School of Electrical Engineering, Computing and Mathematical Science

Reinforcement Learning for Low Probability High Impact Risks

Gareth David Hunt

This thesis is presented for the Degree of
Masters of Philosophy (Computer Science)
of
Curtin University

September 2019

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgement has been made. This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Gareth David Hunt

Acknowledgements

Firstly I would like to thank my supervisors Mihai Lazarescu and Raymond Sheh for their insight and endless patience. I would also like to give thanks to my parents for encouraging me and supporting me through this journey. I would like to the myriad of computer science icons that inspired me to pursue a computer science education. Finally I would like to thank Curtin University giving me to opportunity to create this thesis.

Abstract

In this thesis we demonstrate a method of reinforcement learning that uses training in simulation. Our system is novel in its use of simulation to generate an estimate of the potential reward and risk of transitioning to a failure state of each action as well as a measure of the uncertainty present in both of these. The system generates these estimates in **Danger Training** Mode by seeking out not only rewarding actions but also dangerous ones during the simulated training. During exploitation our system is able to use this knowledge in **Danger Avoidance** mode to avoid risks while getting closer to the goal. We demonstrate that our system outperforms standard Q-Learning in some scenarios. For example, in an environment with a U-shaped safe path, our learner takes a long 'u' shaped path of safe spaces from the start location to the goal whereas standard Q-learning cuts straight across. Unlike our learner, standard Q-Learning is unable to distinguish between high-risk, high reward actions and low risk, medium reward actions. This exposes the standard Q-Learning agent to unnecessary danger and as such our learner has a higher success rate. We also show that Danger Training mode performs a useful function. When our learner starts in Danger Avoidance mode without any prior information about the environment it has no information about where the goal is locate. It will prioritise minimising danger only and perform the same known safe actions repeatedly rather than exploring the environment. Eventually after many runs the random factor of action selection will cause the agent to reach the goal and begin seeking the goal but this represents a period of wasted training time. If instead we train the learner using **Danger Training** mode followed by **Danger Avoidance** mode this is avoided. In **Danger Training** mode the system actively maps out the dangers present within the environment and through this exploration determines the goal location. Next in **Danger Avoidance** mode, now that the goal location is known the agent is able to refine the Q-Value by taking these safe paths to the goal and determining which of them is most advantageous.

Contents

List of Symbols

Our System

s	State
s'	The next timestep's State
S	Set of States
s_F	Failure State
s_G	Goal State
a	Action
A	Set of Actions
$Q(s,a)$	Expected reward
$D(s,a)$	Expected immediate danger of transitioning to a failure state
$C_Q(s,a)$	Certainty of $Q(s,a)$
$C_D(s,a)$	Certainty of $D(s,a)$
$\pi(a\|s)$	Policy
$L_{Training}$	The degree to which exploring $Q(s,a)$ is traded off against $D(s,a)$ during training
Z_Q	A modifier to ensure that an option can still be chosen with some minimum probability regardless of $Q(s,a)$
Z_D	A modifier to ensure that an option can still be chosen with some minimum probability regardless of $D(s,a)$
K	The steepness of a logistic function curve
$x0$	The centre point of a logistic function curve

Comparison Learner

α	Learning Rate
γ	Discount factor
k_+	The degree of risk sensitivity for gains
k_-	The degree of risk sensitivity for losses
l_+	A modifier on the degree of risk adverse behavior for gains
l_-	A modifier on the degree of risk adverse behavior for losses

List of Figures

Chapter 1

Introduction

In many applications, a system will not only have to find a path to reach a goal or achieving an objective but also avoid risks while doing so. These risks may not merely involve a setback but may block the agent from completing its goal entirely, putting the agent in a failure state. These failure states may also incur costs beyond this, such as an expensive robot being damaged.

Thus we need systems capable of simultaneously working towards the goal and managing risk; we may want a robot to take a longer path if the shorter path is over an uneven surface that means the robot may tip over and damage itself. This system needs to be capable of learning about the risks in the environment and responding appropriately, distinguishing between high-risk, high-reward actions and low-risk, medium-reward actions.

The system should also still be capable of managing risk even if factors make the risks difficult to detect. These factors can include low probability, high impact risks and the environment being computationally expensive to simulate leading to a limited amount of training data.

Classical risk sensitive machine learning is often applied to applications such as the stock market in which avoiding losses can be as or more important than achieving gains. However in many applications, particularly those in robotics, risk takes the form of not only setbacks but also complete failures. A robot may not only loose traction and slide down an incline away from the goal, it may fall in a pit and become entirely unable to complete its goal at all.

It should be noted that the consequences of failures are not necessarily limited to the task at hand not being performed; in some applications they may also represent valuable equipment being damaged, lost or destroyed. In certain applications, such as when using very expensive equipment, this may mean that

avoiding failure states is a more important objective than reaching the goal.

The ***aim*** is for the agent to learn how to act in a way that minimises risk while still achieving the goal. Our system considers risk as a chance of transitioning to a failure state in a given action. This is different to most other systems that take risk into account by modelling it as a negative reward. Furthermore our system should also be capable of still performing to some degree when training data is limited leading to uncertain risks.

We present a method for learning to navigate dangerous environments using reinforcement learning that is novel in that it both learns reward and danger separately but also keeps track of the level of uncertainty of both attributes. This allows our system to make choices that avoid both known dangerous states and uncertainty during exploitation. It also allows our system to specifically seek out and map uncertain dangers during exploration.

In this thesis we describe a method of machine learning that takes into account both the risk of failure and the potential of reward. It will also take into account an assessment of the level of uncertainty in each of these attributes. It is capable of choosing actions to effectively explore possible dangers in **danger training** mode and avoid potential dangers while navigating towards the goal in **danger avoidance** mode. In a real application such as a robot navigating in a dangerous environment the exploration would occur in simulation to avoid damaging a real robot before it was run on the physical robot in **danger avoidance** mode.

Chapter 2

Background

The problem of interacting with a dangerous environment in a safe manner has been studied for some time and there's a great deal of work in this area. Furthermore even certain systems not originally designed for avoiding risk can be modified to adopt some degree of risk avoiding behaviour by setting the right reward function.

As such we will cover the literature but focus on the papers most relevant to the task of learning how to operate in a system with risks present.

Given our goal and methodology, we will cover the most relevant approaches in some depth; both those systems that manage risk explicitly and those that manage risk implicitly as a result of their design.

We will start by examining reinforcement learning systems in the context of using their reward functions to account for risks; starting with systems that make no special considerations towards risk and then moving onto describe reinforcement learning systems that model risk explicitly.

2.1 Traditional Reinforcement Learning

Basic reinforcement learning such as non-risk sensitive Q-Learning [Watkins and Dayan (1992), Dearden et al. (1998), Greenwald et al. (2003), Kohri et al. (1997), Ng et al. (2006)] and other reinforcement learning systems not specifically designed for risk management [Sutton (1988), Tsitsiklis and Van Roy (1997), Cichosz and Mulawka (2016)] do not account for risk explicitly but can make some sensible decisions if the reward function is appropriate for the environment.

In basic Q-Learning each state-action pair is assigned a Q-value $Q(s, a)$ which is a measure of the quality of a particular action. After taking a particular action

this value is updated based on the old Q-Value, the reward r, the learning rate α, the discount factor γ and an estimate of the optimal future reward $Max\ Q(s', a)$ such that:

$Q(s, a) = (1 - \alpha) * Q(s, a) + \alpha * (r + \gamma * Max_a\ Q(s', a))$

6 The learning rate and the discount factor are parameters, allowing the user to control the rate at which the agent changes its Q-values based on new data and the degree to which sooner rewards are valued over later rewards.

Thus Q-Learning is capable of choosing actions in a way that takes into account the possibility of failure if the reward function is constructed such that any failure gives a highly negative reward. This means that even if a particular risky action produces positive results most of the time but also results in rare failures these failures will still have a significant effect on the Q-value.

One problem with this approach is that the system will be unable to distinguish between a high risk-high reward action and a low risk-low reward action.

These two actions may, on average, possess equivalent utility in terms of achieving the goal but in many applications they are not equivalent. We don't necessarily want an expensive robot to try driving a short path over risky uneven terrain instead of taking a longer path around, or a stock algorithm to always pick the high risk, high reward investments even if their average yield is highest after the risk of failure is taken into account.

In some applications, reinforcement learning systems can use simulation to gather data in training mode in order to refine their results [Abbeel et al. (2006), Dimitrakakis and Tziortziotis (2013)]. If a problem can be modelled in simulation then the system is able to generate a lot more training data than would be practical to obtain normally in most applications. Furthermore the system is able to transition to failure states and thus learn which actions may lead to failure states without encountering any real world failures.

A type of reinforcement learning that does explicitly account for risk is risk sensitive reinforcement learning.

2.2 Risk Sensitive Reinforcement Learning

A number of other risk sensitive reinforcement learning techniques [Heger (1994), Koenig and Simmons (1994), Mihatsch and Neuneier (2002), Geibel (2001), Geibel and Wysotzki (2005), Huggins and Tenenbaum (2015)] have been established in the literature.

One of the first attempts to create a reinforcement learning system that explicitly handles risk was created by Heger (1994). Their system modifies Q-Learning to attempt to learn the worst case outcome of each action, though negative events too unlikely to occur in training may prevent the system from obtaining a perfect model of all the worst possible outcomes. This allows the system to make extremely conservative choices that avoid most potentially risky actions.

Avoiding all risk, no matter how unlikely, however makes many tasks impossible or extremely inefficient. A robot using this system will never drive across a narrow bridge even if there is no other way across.

The system is also incapable of distinguishing between a low probability-high impact risk and a high probability-high impact risk if the impact is the same. Sometimes when a robot is on the edge of a pit on loose ground attempting to drive away from the pit may cause the ground to give way, causing the robot to end up in the pit; because of that this system will be unable to distinguish between attempting to escape the pit and driving straight into it because the worst case of both is ending up in the pit.

Furthermore in some real world applications all actions carry a chance of complete failure, even if it is low. This will cause the system's model of the action space to slowly be filled with minimum reward, off limits actions as random failures accumulate.

Another early approach to risk management is exponential utility functions as described in Koenig and Simmons (1994). In some ways this approach is a formalisation of the approach to using a traditional reinforcement learner in risk sensitive applications explained in Section 2.1. The rewards from the environment are transformed by an exponential utility function. Thus failures have greater weight than the unmodified rewards would suggest and even relatively rare failures still have a significant impact on the chance of a particular action being chosen.

This approach allows even returns that are inherent to the environment such as an amount of money a stock is valued to use higher penalties at, as opposed to rewards subjectively set by the user adapting the learner to a particular problem such selecting the penalty for a robot falling into a pit, and does so in a consistent manner.

These early systems aside, the first robust and adaptable reinforcement learning system for risk management is described by Mihatsch and Neuneier (2002). Essentially the core insight of this system is that in many applications the variance in the reward is also important and not just the average case or worst case reward.

This allows the user to select the level of risk they are willing to accept depending on the application. This system includes a parameter which allows the user to set the degree to which the system should prioritise avoiding risk over obtaining reward. This allows for the system to not only avoid risky actions but in other applications this allows the user to purposefully seek risky actions in applications where the peak performance is of importance rather than average performance.

However this system assumes that risk takes the form of a continuous cost such as losing money on the stock market as opposed to a discrete failure state such as a robot falling into a pit.

The system described by Geibel (2001) and further developed in Geibel and Wysotzki (2005) is one of the first works specifically designed to learn to avoid failure states. Specifically this system attempts to allow the user to set a limit on the amount of acceptable risk.

They model the world as a MDP with two criteria for each state; a discounted estimate of a state's value and an estimate of a state's immediate risk. The system will then try to find the optimal policy as in standard reinforcement learning but only selecting actions with risks lower than the given bound.

One problem with this approach is that it assumes acceptable risk will remain static for the entire duration of the agent's actions. In many real world applications however risk varies. If a robot spends most of its time traversing a relatively safe environment with a few pits providing some minor risk but at some point must cross a narrow bridge then the acceptable risk bound must be set high enough to accommodate the bridge crossing. This means that the system will not avoid the more dangerous actions during the majority of its time spent in a safer environment as even the most dangerous actions during this phase will be safer than the safest actions during the bridge crossing phase.

Additionally the system only takes risk into account in terms of whether it falls above or below the user's risk threshold. As long as both action's risks are lower than the bound two actions with equal returns are considered equivalent even if one is much safer than the other for no cost.

The system described by Huggins and Tenenbaum (2015) combines risk sensitive learning with regret sensitive learning as described in Jaksch et al. (2010) in which systems attempt to set bound on the degree of suboptimality or regret the agent is willing to risk.

Attempts have also been made to model and replicate human risk taking behaviour in a reinforcement learning context such as Shen et al. (2014), which

we later used as a comparison learner. This work modifies the risk sensitive Q-learner outlined in Mihatsch and Neuneier (2002) such that the system alters how much risk it is willing to accept dynamically at runtime depending on the results it is obtaining.

The system will respond to choose high risk-high reward actions if it consistently obtains positive results and low risk-low reward actions if it consistently obtains negative results. This means that the model can be systematically wrong in some way (eg in training it failed to explore some kinds of risky areas) and the system will correct itself relatively quickly. Or if the environment has changed (eg if it has recently rained and now all the surfaces are riskier than in training) it can change its behaviour without relearning each individual state-action.

They demonstrate that the system performs empirically similar to humans on problems like a simple stock market game. Furthermore they compare this model to human MRI brain scans while subjects are tasked with decision making and show some similarities in the way the model makes decisions and how actual humans make decisions. As a system based on Mihatsch and Neuneier (2002) it shares some of its problems in practical use.

A more recent example is described in Van Moffaert et al. (2015). Like Mihatsch and Neuneier (2002) this learner considers not only the average result of an action but also the variance of the results. However unlike that previous system this work does not use the variant of the result to weight the Q-value but instead uses Pareto Q-learning to learn how to achieve the best result in for both average result and variance separately and then generates a Pareto optimal policy that settles on the best trade off between these two goals.

There has also been work in risk sensitive inverse reinforcement learning [Majumdar et al. (2017)] which seeks to better understand an environment by observing agents acting within it and taking into account their risk management decisions.

2.3 Vector Fields

Vector fields represent the environment as a series of vectors. These vectors point towards desirable locations in the environment such as the goal and away from undesirable locations such as dangerous spaces. This has been applied to making robots navigate complex environments for some time [Borenstein and Koren (1989), Borenstein and Koren (1990)].

However vector fields differ from the kind of system we want for our use case in several ways:

- Vector fields assume you already have the vector field for the environment available. Though our system requires the ability to simulate the environment this does not necessarily imply exact knowledge of the value or danger of each state without conducting simulated trials. Thus we want our system to have the ability to explore the environment efficiently rather than having to conduct multiple simulations of every action in ever location within the environment to generate the vector field.

- We want our the system to consider not only its knowledge about the risks present in the environment but also the system's level of certainty in its knowledge of these risks, allowing the system to make safe choices based on limited data.

2.4 Learning by Imitation and Behavioural Cloning

Another potential approach to risk management via reinforcement learning would be to learn from a human expert demonstrator. Learning systems that Learning by Imitation [Michie et al. (1990)], Behavioural Cloning [Bain and Sammut (1996), Michie (1998) Kadous et al. (2006), Brown and Sammut (2013)] and other approaches that use human generated training data [Atkeson and Schaal (1997), Argall et al. (2009), Brys et al. (2015)]. This approach uses a human expert to generate training data on the given task under the assumption that a human would factor in any relevant information into their decisions including, most importantly for our purposes, any risk management concerns.

The fundamental idea behind Learning by Imitation is that, on average, a human expert will make the correct decision in any given situation. Thus by taking an action that mimics the action taken by a human expert in every state the agent is able to replicate this behaviour. Additionally by using the average action any non-systematic mistakes by the human experts will be ignored resulting in performance that can exceed the human demonstrator.

The reason why we cannot necessarily have the human expert simply explain his methodology to the system architect to replicate in software is that many of the kinds of behaviours we want our systems to replicate are partially or fully

subconscious, particularly in robotics use cases. This means that the human experts can only describe their process in, at best, vague subjective terms such as "you hit the accelerator when you feel as though you have enough grip" unsuitable for translation into software. For this reason one type of Learning by Imitation known as Behavioural Cloning is built to model subconscious actions by human experts in particular.

One potential pitfall of using human expert demonstrators is that they will often fail to explore failure neighbouring states [Sammut et al. (1992)]. Humans, for instance, won't choose to drive a robot along the edge of a pit if the path is wide enough to avoid it.

In theory this isn't a problem as the system will replicate the behaviour of the human expert demonstrators and never arrive in failure adjacent states but in practice real world conditions will occasionally result in the system encountering these states regardless. Thus if the system ends up in one of these failure neighbouring states despite its best efforts it may lack sufficient training data in order to make an informed decision on how to avoid imminent failure.

To a lesser extent more traditional reinforcement learning systems may encounter a similar problem. As soon as region is characterized as having a low expected reward due to the possibility of entering a failure state the agent of most traditional reinforcement learning systems will be less likely to this region even if the system's parameters are set to prioritise exploration over exploitation.

A recent paper by Santara et al. (2017) attempts to approach Imitation Learning in a way that takes risk into account. It considers risk in a similar way to risk sensitive reinforcement learning approaches, modelling action cost rather than explicit failure states. However while this system attempts to choose safer actions with the information it possesses, taking into account the bias present in human expert demonstrator derived training data, it does not attempt to gain more data to fill any potential gaps in the system's understanding.

A system described in Sheh (2010) attempts to solve this problem by building upon Behavioural Cloning an explicit understanding of failure states. Distinct from prior Fault Detection systems such as Chiang et al. (2001) which attempt to learn which kinds of sensor outputs correspond to a failure state this system instead attempts to learn which actions are more likely to result in the agent transitioning to a failure state.

In the Situation-Action model they describe a system in which the system weights the discounted reward against the probability of transitioning to a failure state when taking an action. The human expert derived training data is supple-

mented with training data derived from an autonomous demonstrator running in simulation. This demonstrator uses an A* search that uses the ability to rewind a simulated environment to explore multiple possible paths in a single run.

This not only provides the system with much more training data than could be practically obtained from human experts but the autonomous demonstrator is also intentionally started in failure adjacent states in some runs in order to gain more data on how to avoid failure states. This system works well for some applications but isn't ideal for applications in which simulating more training data is particularly computationally expensive or applications in which, rather than explicit pits that must be avoided, each action involves a low amount of risk making the signal of less or more risky actions more difficult to distinguish from the noise.

2.5 Summary

Traditional reinforcement learning will be affected by risk insofar that the average expected value of an action will be decreased by a failure. However it will be unable to distinguish between a low risk-low reward action and a high risk-high reward action.

Risk sensitive reinforcement learning is able to detect the difference between low risk-low reward and high risk-high reward actions. However these systems either consider risk as a negative reward and not a separate attribute or only check if the risk is below a given bound.

Systems that learn from human experts such as Learning by Imitation and Behavioural Cloning systems are able to learn from the risk management decisions of their human demonstrators. However most systems will not have much data in failure adjacent states due to the problem described in *Learning to Fly* by Sammut et al. (1992) in which human experts avoid failure states to the extent that human expert trained agents have little information on failure adjacent states. The Behavioural Cloning System by Sheh (2010) avoids this by considering risk as a separate attribute and purposefully starting a simulated demonstrator in failure adjacent states in order to obtain more data on these dangerous regions.

Our work will, similar to the system described by Sheh (2010), consider the possibility of a failure when taking any particular action and avoid the problem of not exploring failure adjacent states described by Sammut et al. (1992). In addition to this however it will be sensitive to slight differences in risk between

actions and possess the ability to make reasonable decisions even when training data is limited by computation possessing a high cost.

Chapter 3

System Outline

In this chapter we will describe our system and the simulated environment we will conduct our experiments within. First we will show the definitions used in our experimental set up and system then we will outline the pseudo code and give a more detailed explanation.

Next we will describe one of the two comparison systems we will use, the system outlined by Shen et al. (2014). The other comparison learner we will use is standard Q-Learning Watkins and Dayan (1992). We will compare these comparison learners against our system in both theoretical examples and in later chapters how it quantitatively performs in real experiments.

Finally we will show how this system will act in contrast with the comparison learner in a range of theoretical scenarios.

3.1 Our System

Our system is designed to take risk into account while also ensuring that failure adjacent states are explored and that the system is capable of operating in a limited training data environment with difficult to detect differences in risk between various actions. It does this by taking into account not only the current risk estimate of each action but also the uncertainty of this risk. This means the system is able to choose a known relatively safe action over actions that have not been attempted enough times to quantify how risky they are.

Additionally our **danger training** function avoids the problem described in Sammut et al. (1992) while also prioritising important data collection when training data is limited. We will now show how this system, and the environment it operates in, is defined.

Definitions

Our experimental set up consists of a World W that is a gridworld of width w by length l squares. This world represents an environment in which failure is not necessarily detrimental during training but must be avoided in exploitation. For instance an environment that can be simulated but at high computation cost, limiting the amount of training data.

The world can be described as a Markov Decision Problem (S, A, P, R) as defined in Bellman and Kalaba (1957) where:

- The set of states S is the set of gridsquares $s \in \{s_{11}, s_{12}, ..., s_{lw}\}$ (including the start state $s_0 \in \{s_{11}, s_{12}, ..., s_{lw}\}$ and the goal state $s_G \in \{s_{11}, s_{12}, ..., s_{lw}\}$) plus a failure state $s_F \notin \{s_{11}, s_{12}, ..., s_{lw}\}$.
- The set of actions A is composed of $a \in \{Up, Down, Left, Right\}$.
- The state transition distribution $P(s'|s, a)$ is the probability of transitioning to state s' after taking action a in state s.
- The reward function R. It is 0 for all states except the goal state s_G where it is positive and the failure state s_F where it is negative.

In this environment one gridsquare is the start state s_0, $s = s_0$ at time step $t = 0$. Another gridsquare is the goal state s_G. The agent will continue to take actions in the world until it reaches either the goal state s_G or the failure state s_F.

Each action a taken in each state s has a probability associated with it, $P_M(s, a) \in \{0.1, 0.9\}$ (with a uniform discrete distribution with increments of 0.1) of the agent moving in the intended direction and $(1 - P_M(s, a))$ of moving to another of the 3 adjacent states (all 3 equally probable, collisions with the edges of the world result in the agent not moving from its current state). On arriving at the next state and before taking the subsequent action, the agent has a probability $P_D(s) \in \{0, 0.35\}$ (though in some experiments it is set to 0 for certain safe states) of making a further transition to the failure state s_F and a probability $(1 - P_D(s))$ of staying in that state. Neither the $P_M(s, a)$ value nor the $P_D(s)$ value are visible to the agent.

Each state has a danger value $P_D(s)$ where $0.03 \leq P_D(s) \leq 0.35$, uniformly distributed, this represents the probability that the system will instantaneously transition to the failure state s_F after arriving without allowing any further actions. As such it models a landscape in which each location has a definite risk

associated with it as opposed to an environment in which each action in a given location has a definite risk. This allows us to quantify exactly what areas in the environment are considered safe and which are considered risky. States with $P_D(s) \leq 0.15$ will be known as safe states whereas states with $P_D(S > 0.15)$ will be known as dangerous states. This allows us to assess how well each learner models its environment's risks in an easily human readable format.

At each time step the agent will choose an action a. After taking the action the agent will have a chance of arriving at any of the 4 adjacent gridsquares (or arriving back in the same state if the gridsquare is on the edge of the world) or the failure state s_F.

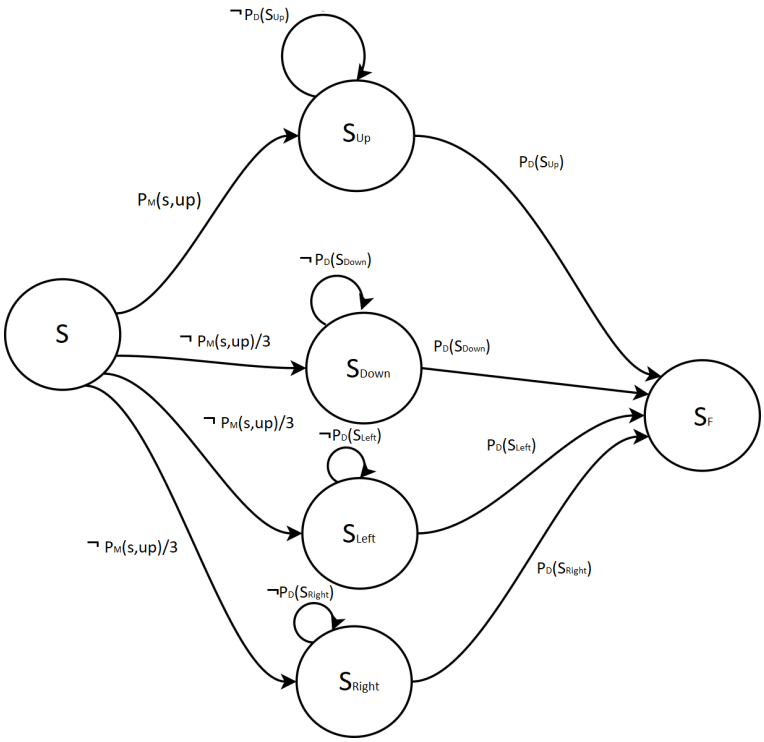

Figure 3.1: The Markov State Diagram of the up action.

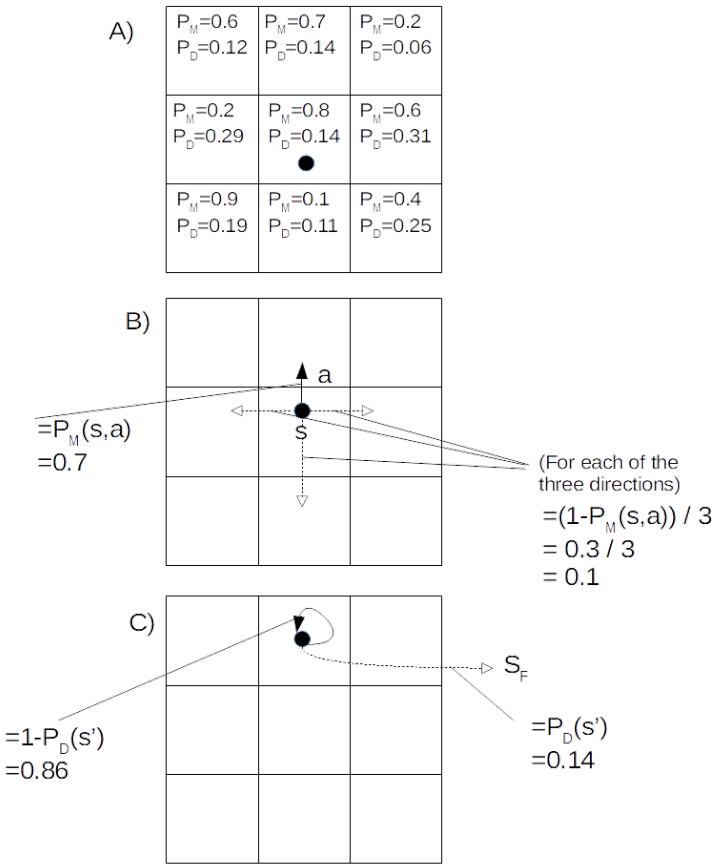

Figure 3.2: The Physical diagram of the up action. The black dot represents the agent's current location. Arrows represent possible transitions, with the solid black arrow being the most likely one.

The Markov State Diagram for choosing the Up action is shown in Figure 3.1. A diagram showing how this same movement works physically within the environment is shown in Figure 3.2.

Step A of the physical diagram shows the agent in the environment before taking the move action; each of the squares within the environment already has an associated P_M and P_D value that is invisible to the agent. In step B of the physical diagram there is a $P_M(s, a)$ probability that the agent will successfully move in the intended direction where $P_M(s, a) = 0.7$ as that is the P_M value of the state the agent is attempting to move into. If the agent does not move in the intended direction then it has an equal chance of transitioning to any of the other adjacent states, thus there is a $(1 - P_M(s, a))/3$ probability that the agent will transition to a non-intended adjacent state.

In step C of the physical diagram the system now has a $P_D(s')$ chance of

instantaneously transitioning to the failure state s_F without allowing the agent to select another action. The value of $P_D(s')$ is 0.14 as that is the value of the state the agent transitioned to in the previous stage, if the agent had accidentally moved downwards then the P_D value would be 0.11 as that is the P_D value of that gridsquare. If the agent has not transitioned into a failure state and has not reached the goal state s_G the agent selects another action and repeats the process.

However note that $P_M(s, a)$ and $P_D(s)$ are simply used in order to make the dangers present in the environment and how various learners respond to them clearer for a human readers to understand. The effect of both values on the agent after taking an action is equivalent to standard gridworld probabilities of moving to each adjacent state and the failure state. By combining the effects of both values on the agent a standard transition probability could be generated, for instance the probability of moving in the intended direction is $P_M(s, a) * (1 - P_D(s'))$ where $P_D(s')$ is the danger value of the state that the agent is attempting to move to. As such other reinforcement learning systems such as standard Q-Learning or the system described by Shen et al. (2014) are not artificially disadvantaged by this approach as it is equivalent to other gridworld with risk implementations, merely with a different set of transition probabilities.

The agent keeps track of 4 attributes for each state-action pair:

- $Q(s, a)$ - The expected reward of the action a in state s. Can either be a heuristic in heuristic mode or the expected discounted reward as determined by another learner (eg Q-Learning) in secondary learner mode. We will outline heuristic and secondary learner modes in Section 3.1.1, note that in heuristic mode $Q(s, a)$ does not represent expected discounted sum of future reward.

- $D(s, a)$ - The expected immediate danger of transitioning to a failure state when taking the action a in state s.

- $C_Q(s, a)$ - An estimate of the level of certainty of $Q(s, a)$. Where $0 \leq C_Q(s, a) \leq 1$ with higher values representing greater certainty.

- $C_D(s, a)$ - An estimate of the level of certainty of $D(s, a)$. Where $0 \leq C_D(s, a) \leq 1$ with higher values representing greater certainty.

These 4 values are taken into account in different ways to create a policy $\pi(a|s)$ $\forall s \in S$ depending on whether the policy is currently in **danger training** or **danger avoidance** mode.

$L_{training}$ is a parameters that determines how $Q(s, a)$ and $D(s, a)$ are weighted against each other in training mode and can be set to anything within the $0, 1$ range. If $L_{training}$ is set to 1 $Q(s, a)$ and $D(s, a)$ will be weighted equally. If $L_{training}$ is set to less than 1 then exploring states with higher $D(s, a)$ values will be prioritized over states with higher $Q(s, a)$ values, the extent to which higher $D(s, a)$ values are prioritized being determined by how low the $L_{training}$ value is.

We define a conservative choice as one that minimises $D(s, a)$ and $C_D(s, a))$. This means that it is a relatively safe choice that avoids both known and unknown risks to the greatest degree possible.

3.1.1 Heuristic Mode and Secondary Learner Mode

Our system has two different modes for learning $Q(s, a)$ values; heuristic mode and secondary learner mode. In heuristic mode $Q(s, a)$ values are determined by a simple heuristic such as distance to the goal in the result stat. Reaching the goal is still highly positive and transitioning to a failure state highly negative but other states are assigned a less positive reward based on how close they are to the goal. This allows the learner to account for the $P_M(s)$ values of different states but the learner mainly focuses on the $D(s, a)$ values for more complex behavior than a simple greedy approach.

In secondary learner mode the $Q(s, a)$ value is determined by a secondary conventional reinforcement learning system. In our experiments using secondary learning mode we use Q-Learning as the secondary learner, thus $Q(s, a)$ represents expected discounted sum of future reward. .

Originally we performed experiments using heuristic mode alone, reasoning that the system's ability to detect and avoid danger would produce an effective policy despite the greedy approach in determining which actions possessed greater positive utility. However several flaws in this approach became apparent based on reviewer comments. The most significant is that the learner was unable to recognize the value of certain kinds of counterintuitive actions such as temporarily moving away from the goal in order to move around a dangerous region before continuing toward the goal.

By using an established reinforcement learner such as Q-learning and back chaining the $Q(s, a)$ values secondary learner mode is able to generate a more accurate $Q(s, a)$ value which better represents each action's utility. If the agent more frequently reaches the goal by going around a dangerous region than by taking the direct path through the dangerous region and potentially failing then

the $Q(s, a)$ value of moving around the dangerous region will eventually become higher.

However heuristic mode does have one advantage over secondary learner mode using Q-learning, as each action is independent the path that the agent follows during each training run does not impact the policy given that the same set of state-action pairs are visited the same number of times. This makes the off-policy **danger training** mode learning approach ideal as this mode prioritizes learning the most important state-action pairs over any particular order.

Secondary learner mode using Q-learning on the other hand will most quickly learn accurate $Q(s, a)$ values when the best paths to the goal are taken in order for the $Q(s, a)$ values to back chain back to the start square. **Danger training** mode will only occasionally follow the ideal path given this mode's preference for exploring dangerous states over safe states. On the other hand the off-policy **danger training** mode will still be the most effective way to learn the $D(s, a)$ values even in secondary learner mode. One potential solution is to split the training runs between training runs performed in **danger training** mode and training runs performed in **danger avoidance** mode before deploying the policy in the real environment.

3.1.2 Pseudocode

Algorithm 1 Action Selection - Runtime

procedure ACTION SELECTION - RUNTIME
    for all Actions **do**
        $\pi(a|s) = (Q(s, a) * C_Q(s, a) * C_D(s, a))/D(s, a)$
    Randomly choose action in proportion to each action's $\pi(a|s)$

Algorithm 2 Action Selection - Training

procedure ACTION SELECTION- TRAINING
    for all Actions **do**
        $\pi(a|s) = (1-C_D(s, a))+(1-C_Q(s, a))+((1+Z_D)-C_D(s, a)*L_{training}*D(s, a)) + ((Z_Q - C_Q(s, a) * Q(s, a)) * (1 - L_{training})$
    Randomly choose action in proportion to each action's $\pi(a|s)$

Algorithm 3 Learning Stage

1: **procedure** LEARNING STAGE
2: Determine s'
3: Determine $Q(s'|a)$
4: **if** $s' = s_F$ **then**
5: $Q(s', a) = 0$
6: $D(s, a) = \sum Failures_{s,a} / \sum Attempts_{s,a}$
7: $C_Q(s, a) = 1/(1 + \exp^{-K_{pos} * (Attempts_{s,a} - x0_{pos})})$
8: *Failure Confidence* $= 1/(1 + \exp^{-K_{neg} * (failures_{s,a} - x0_{neg})})$
9: *Skeptical Confidence* $= 1/(1 + \exp^{-K_{skep} * (Attempts_{s,a} - x0_{skep})})$
10: C_D*(s,a) = Max (Failure Confidence, Skeptical Confidence)*

Explanation

Action selection and the learning occur in parallel, with the learning observing the results of the action selection.

$Q(s, a)$ is initially learned off policy so initial values will be biased by the first couple of returns when the state has only been visited infrequently. However initially $C_Q(s, a))$ will be low so the agent will generally still avoid choosing an action with values biased by early returns that do not properly represent the average return in **danger avoidance** mode.

$D(s, a)$ is based on state-actions despite the failure state transitions only being based upon the state s'. $D(s, a)$ also takes into account the probability of the correct move causing the system to transition to an unintended location and then randomly transitioning to the failure state based on that unintended state's value of $P_D(s')$.

In **danger training** mode the system will attempt to maximise $Q(s, a)$ and $D(s, a)$ while minimising $C_Q(s, a)$ and $C_D(s, a)$. The system will attempt to determine both which actions are particularly beneficial and which actions are particularly dangerous to perform by choosing actions that are high in either attribute that also have low certainty.

The **danger training** mode action selection can be split into two parts:

- First the state-action's uncertainty is taken into account with $(1-C_D(s,a))+(1-C_Q(s,a))$. As these values are between 0 and 1 taking the opposite of these values represents the uncertainty of these qualities. This is taken into account independently of $Q(s,a)$ and $D(s,a)$ first as it is valuable to explore all sufficiently unexplored state-action pairs even if they do not appear to be dangerous or lead the agent closer to the goal as these values will not be based on meaningful data until the state-action is attempted a number of times.

- The second part takes $Q(s,a)$ and $D(s,a)$ into account with $((1+Z_D)-C_D(s,a)) * L_{training} * D(s,a) + (Z_Q - C_Q(s,a)) * Q(s,a) * (1-L_{training})$. The extent to which these values are taken into account depends on their respective certainty values, $C_D(s,a)$ and $C_Q(s,a)$. Actions that appear to bring the agent reliably closer to the goal or cause the agent to encounter risk that have not yet been confirmed are more important to explore than actions that have high $Q(s,a)$ and $D(s,a)$ but have already been explored to a high degree of certainty.

However as certainty approaches 1 we do not want $Q(s,a)$ and $D(s,a)$ to no longer be taken into account. As such Z_Q and Z_D are added to ensure that these values always effect the change of a given action being chosen, albeit in a discounted fashion as uncertainty approaches 0. Currently they are both set to 0.1 based on experimental results.

In **danger avoidance** mode the system will attempt to maximise $Q(s,a)$, $C_Q(s,a)$ and $C_D(s,a)$ while minimising $D(s,a)$. Actions that move the agent toward the goal while having high levels of certainty are chosen while actions that are dangerous are avoided. This avoids both known dangers and potential unknown dangers.

Note that in both **danger training** mode and **danger avoidance** mode the sum of all the $\pi(a|s)$ values will not add up to one and as such are not themselves probabilities. Rather the probability of each action being chosen is determined by their relative values - the probability of each action being chosen is proportional to $\pi(a|s)/\sum\pi(a|s)$.

However local minima of $D(s,a)$ values are possible under certain conditions. If a given region is sufficiently safer than all its surroundings and leaving the safe region and entering a more dangerous state is required to reach the goal then the

value of choosing to leave the safe region and progress towards the goal may never exceed the value of staying within it. For this reason the **danger avoidance** mode choices are also chosen probabilistically rather than deterministically.

The $C_Q(s, a)$ and $C_D(s, a)$ are calculated with a logistic function. These functions will always return a value between 0 and 1 based on the number of times an action has been attempted or the number of failures the system has encountered. The K values determine the steepness of the curve and $x0$ determines its centre point. These values are adjusted to control how many examples are necessary to obtain a given certainty value. In effect they control how quickly the system increases in confidence.

The purpose of skeptical confidence is to handle actions that have very little risk such that even with multiple attempts the learner never transitions to a failure state. Normally the learner's confidence in the $D(s, a)$ value depends on how many times it has encountered a failure attempting a given action. However if the learner attempts this action many times and still encounters few or no failures it should still eventually grow more confident in the $D(s, a)$ value. As such in addition to the negative confidence value which increases as the learner transitions to a failure state after attempting the action, we use a skeptical confidence value which increases as the action is attempted more times but at a much slower rate than either the $C_Q(s, a)$ value or negative confidence value.

We then determine the $C_D(s, a)$ value by taking the maximum of the negative confidence and the skeptical confidence. As the negative confidence value increases at a faster rate than the skeptical confidence value it will become the $C_D(s, a)$ value except when the learner has rarely or never transitioned to a failure state when attempting a given action, in which case the skeptical confidence value becomes the $C_D(s, a)$ value.

The ideal values for these parameters will vary depending on the application - an application with difficult to detect failures that only occur rarely should have a steeper slope for failure confidence and a shallower slope for skeptical confidence than an application in which all risks are relatively common.

In this application one set of sensible values for these parameters is to set the K values such that $K_{pos} = 0.05$, $K_{neg} = 0.1$ and $K_{skept} = 0.025$ and set the $x0$ values such that $x0_{pos} = 10$, $x0_{neg} = 50$ and $x0_{skept} = 100$.

As shown in Figure 3.3 this means that $C_Q(s, a)$ will approach 1 after around 100 data points, Failure Confidence will approach 1 after around 20 failures and skeptical confidence will approach 1 after around 200 data points.

By selecting the maximum of a steeper curve based on failures and a more

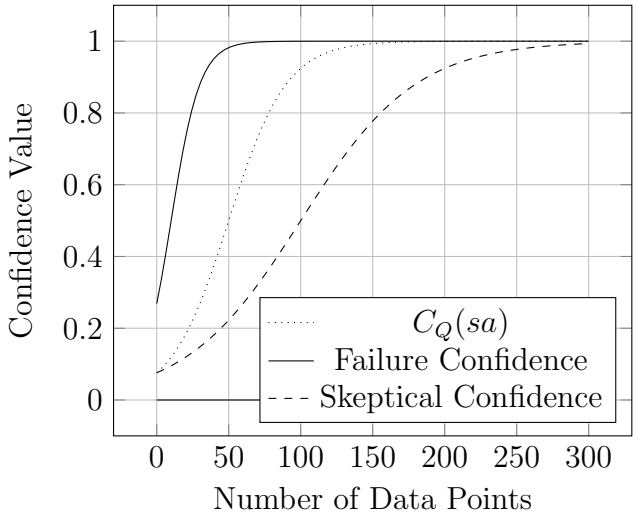

Figure 3.3: A plot of the logistic equations governing $C_Q(s, a)$ and $C_D(s, a)$.

gradual curve based on number of attempts the system is able to best handle the risk uncertainty of both dangerous and safer actions. If the action is dangerous the number of failures will rise rapidly and the failure curve will dominate, allowing the system to quickly learn of a dangerous action. If the state is safe and relatively few failures occur the system will still become more certain of the safety of the state as shallower attempt based curve dominates.

Line By Line Explanation

Given this understanding of the system we can now run through an explanation of how each line of the psuedocode works.

Runtime Mode Action Selection:

For all actions:

$\pi(a|s) = (Q(s, a) * C_Q(s, a) * C_D(s, a))/D(s, a)$

We determine how beneficial each of the available actions is to the agent, taking into account both positive and negative factors. The higher the potential reward ($Q(s, a)$) and the more information we have about the potential action ($C_Q(s, a)$ and $C_D(s, a)$) the higher this average benefit will be, the higher the danger ($D(s, a)$) the lower the average benefit will be.

Randomly choose action in proportion to each action's $\pi(a|s)$

Each of the four possible directions will have a $\pi(a|s)$ value. The larger a particular direction's value compared to the other direction's values the more likely that direction is to be chosen. For instance if moving up has a value of

5, moving right has a value of 3, moving left a value of 2 and moving down a value of 1 (for a combined value of 10) then the agent has a 5/10 or 50% chance of choosing to attempt to move up. This does not mean the agent will always succeed in moving up as this does not take into account the $1 - P_M(s, a)$ chance of transitioning to a state in an incorrect direction or the chance of transitioning to the failure state.

Training Mode Action Selection:

For all actions:

$\pi(a|s) = (1 - C_D(s, a)) + (1 - C_Q(s, a)) + (Z_D - C_D(s, a) * L_{training} * D(s, a)) + (Z_Q - C_Q(s, a) * Q(s, a)) * (1 - L_{training})$

In this line determine how to prioritize the information each action can give us by considering both how confident we are in the information gathered to date ($C_Q(s, a)$ and $C_D(s, a)$) , our current estimate of danger ($D(s, a)$) and our current estimate of the reward ($Q(s, a)$).

The importance of the first is the most obvious; it is more important to explore actions for which we are not yet confident will produce good or bad results than further confirming those actions we are already confident about. This is accomplished by both the first part of the code $(1 - C_D(s, a)) + (1 - C_Q(s, a))$ which ensures this factor is still taken into account when our estimates of reward or danger are set to zero and also $C_D(s, a)$ and $C_Q(s, a)$ acting as multipliers in the next two parts of the this line of code.

We want to prioritize exploring actions that appear more dangerous to either confirm their level of danger to ensure the agent the agent does not attempt them or to reveal that early failures encountered when taking this action were flukes and this action may be instead only pose a moderate risk. The degree to which this is prioritized over exploring rewarding states is determined by $L_{training}$. This is accomplished by the $(Z_D - C_D(s, a) * L_{training} * D(s, a)$ part of this line of code.

We also want to prioritize exploring actions that appear rewarding as these actions may be the part of the best path to the goal and should be explored to either reveal any potential dangers or confirm they are safe for the agent to traverse. This is accomplished by the $(Z_Q - C_Q(s, a) * Q(s, a)) * (1 - L_{training})$ part of this line of code.

Both Z_Q and Z_D should be set to values slightly above 1 (eg 1.1) to ensure that will give some value to choosing an action even when C_Q and C_D are high.

Randomly choose action in proportion to each action's $\pi(a|s)$

This is the same as in the runtime mode's action selection; the learner chooses one of the four directions to move in with each action's $\pi(a|s)$ value increasing

the chance of it being chosen.

Learning Stage:

Determine s'

Determine $Q(s'|a)$

First we need to determine the agent's current state after taking the action. Next we determine the reward value of the current state.

This is the only step that differs between heuristic mode and secondary learner mode. In heuristic mode this value is based on the distance to the goal with closer distances having higher values. In secondary learner mode a secondary reinforcement learning algorithm is used to determine this value. In our secondary learner mode examples we use the Q-learning equation here to determine this value.

If $s' = s_F$

$Q(s', a) = 0$

We detect whether the agent has encountered any failure condition. If so any reward value the agent has encountered is void, the agent is considered to be in a zero reward failure state.

$D(s, a) = \sum Failures_{s,a} / \sum Attempts_{s,a}$

We update our $D(s, a)$ value with information about whether we encountered a failure state in our latest attempt of this action.

$C_Q(s, a) = 1/(1 + \exp^{-K_{pos} * (Attempts_{s,a} - x0_{pos})})$

In this line we use a sigmoid function to turn the number of times we've attempted an action as an estimate of how confident we are that the $Q(s, a)$ value is accurate. A sigmoid function will always give us a $C_{Q(s,a)}$ value between 0 and 1.

The S shape of the function can serve as a basic confidence model. First the confidence value increases slowly as one or two examples may be flukes not representative of the average action. Then as a body of evidence is established we increase our confidence more quickly. Finally we slow our increase in confidence as it approaches 1 as no finite amount of evidence will allow us to be entirely confident in our $Q(s, a)$ value's accuracy.

The slope of the function and its midpoint are determined by the K_{pos} and $x0_{pos}$ values respectively and ideal values are likely application specific.

Failure Confidence $= 1/(1 + \exp^{-K_{neg} * (failures_{s,a} - x0_{neg})})$

If we encounter a number of failures early in exploration we can confidently conclude an action might be dangerous early on. This line uses a sigmoid function like the prior example to turn the number of failures encountered when trying a

particular action into a confidence value between 0 and 1.

Skeptical Confidence $= 1/(1 + \exp^{-K_{skep}*(Attempts_{s,a} - x0_{skep})})$

If we only keep track of failures we will never increase our confidence in the level of danger posed by actions in which we encounter few or no failures; leading to our system never trusting safer actions. As described in the prior section skeptical confidence is used to ensure that even if the agent never transitions to a failure state after attempting a particular action we will still slowly increase the $C_D(s, a)$ value. As such we also use a sigmoid function to turn the number of attempts of an action into an estimate of our confidence in $D(s, a)$ as we did to estimate our confidence in $Q(s, a)$ earlier. However this function should always be a shallower slope than the prior two examples that takes a larger number of total attempts/failures than the previous two functions to reach a certain level of confidence.

C_D*(s,a) = Max (Failure Confidence, Skeptical Confidence)*

By taking the higher of these two metrics our learner is capable of either quickly coming to the conclusion that an action might be dangerous or slowly coming to trust a safe action.

3.1.3 System Performance

Based on the analysis performed in Koenig and Simmons (1993) we have determined that our learner in the gridworld environment we laid out in Section 3.1 has a big O complexity similar to that of traditional Q-Learning, $O(N^2)$. However in other applications and environments the complexity of both our system and Q-Learning can be as high as $O(en)$ though this can be practically limited to N^3 with initial values and task representation. In this formulation N represents the size of the state space and e represents the total number of actions available in every state.

We compared the performance of our system to the comparison learner. This learner, as a system based on Q-Learning, also has a Big O complexity of $O(N^2)$ within a gridworld and as such should scale similarly. Both systems were recorded on a old computer with an Intel Pentium E5400 processor and 4 Gb of memory. For this experiment both systems had all diagnostics, all visualizations and all I/O disabled except where necessary to the operation of the system. An average of the time required for the simple safe path, random safe path, simple danger region, random danger region and the 4 random worlds later used in Section 4 was taken to produce these numbers. As these worlds are all 10 by 10, $N = 100$.

Learner	Mean Time Taken
Our Learner	72.5 seconds
Comparison Learner	106.1 seconds

Table 3.1: The average time taken to perform 10000 runs for our learner compared to the comparison learner on 8 test worlds.

When I/O and diagnostics were enabled however both learners performed significantly slower and these numbers cannot be taken as the time practically required perform the simulations to reproduce these results.

3.2 Comparison Learner

In our experiments we use two different learners as a comparison against our system. The first is standard Q-Learning as described in Watkins and Dayan (1992), serving as a well known algorithm with well known properties to compare our learner against. The second comparison learner on the other hand is less well known and requires a more detailed introduction.

The second comparison learner is an implementation of the risk sensitive reinforcement learning system described in Shen et al. (2014) which in turn is based upon Mihatsch and Neuneier (2002). This is a form of Q-Learning modified to change its behavior depending on the risks it encounters. We have chosen to use this system as a second comparison against ours.

Systems that use Learning by Imitation or Behavioural Cloning in the design such as the system described by Sheh (2010) are not suitable for direct comparison as they do not generate training data in the same way but rather rely upon the input of human experts for at least part of their training data set. We require a learner that can be limited to the same conditions as our system: a limited set of exploratory runs within the world.

Furthermore by selecting its risk-sensitive choices in a human like fashion (as Shen et al. (2014) attempts to replicate human risk taking behavior) it provides a way to test our system against a rough simulation of a human's performance without having gathered thousands of human runs. It should be noted that this is better seen as an average human's attempt at these kinds of problems and may not reflect the results obtained by a human expert trained on the best strategies available for a given problem and environment.

A system described in a paper by Santara et al. (2017) may have also served

as a suitable comparison as this system also outlines a reinforcement learning system that avoids dangerous actions but it was published when our comparison experiments were already being performed.

These systems do not generally consider failure or risk as a separate quantity to reward but instead model a risky action as one with a potentially negative reward. This works well for modelling applications such as potential stock market losses but is less well suited for problems such as potentially damaging an expensive robot by driving it into a pit. Additionally these systems do not consider the level of confidence the system has in these risk measurements therefore will make worse decisions than is possible with all the available data.

3.2.1 Pseudocode

Action Selection Stage:

Algorithm 4 Action Selection

procedure ACTION SELECTION
    for all Actions **do**
        $P(a|s) = e^{Q(s,a)} / \sum(e^{Q(s,a)})$
    Randomly choose action in proportion to each action's $P(a|s)$

Algorithm 5 Learning Stage

procedure LEARNING STAGE
    if $r_t \geqslant 0$ **then**
        $utility(r_t) = k_+ * r^{l+}$
    else
        $utility(r_t) = k_- * -r^{l-}$
    $\alpha = 1/n_{visits(s,a)}$
    $learnedvalue = utility(r) * (r_t + \gamma * max(Q(s_{t+1}, a)) - Q(s, a)) - x0$
    $Q(s_t, a) = Q(s, a) + \alpha * learnedvalue$

In our experiments, we use the same gridworld environment to test both the comparison learner as well as our system. Definitions pertaining to the environment itself such as state s and action a also apply for the comparison learner.

The utility of an action is an estimate of the usefulness of a particular action based on the reward and the level risk the system is willing to take.

The learning rate α is a variable that determines the extent to which new results are prioritized over prior knowledge when determining the new Q-value. It decreases as the agent visits the state more times, representing greater evidence being required to change the current estimate of the value of a particular action as the agent becomes more certain.

The discount factor γ is a parameter that determines the extent to which future rewards are prioritized over current rewards. The system has separate parameters for both positive and negative risk sensitivity, k_+ and k_- respectively. They have values of $-1, 1$ and determine whether the risk function is convex or concave.

The modifier values l_+ and l_-. They determine the extent to which positive and negative rewards are valued in the system. Both l_+ and l_- are generally set to values less than 1, which makes the system risk adverse when gains are being made and willing to take risks when losses are being incurred already.

3.2.2 Line By Line Explanation

Action Stage

$P(a|s) = e^{Q(s,a)} / \sum(e^{Q(s,a)})$

This is a standard softmax action selection. Each action is given a value between 0 and 1 in proportion to its Q-Value. This value is the probability of that action being taken.

Learning Stage

$If\ r_t \geqslant 0$

$utility(r_t) = k_+ * r^{l_+}$

$else$

$utility(r_t) = k_- * -r^{l_-}$

Humans act differently depending on whether they're making a profit or avoiding loss. This replicates that behavior by determining the utility of an action with a different set of parameters depending on whether it's positive or negative.

$\alpha = 1/n_{visits(s,a)}$

As with many standard Q-Learning approaches the learner decreases the learning rate as we try a particular state-action more times. This means that the learner slowly becomes more confident in a given Q-Value for a particular state-action pair instead of altering it dramatically after every action.

$Q(s_t, a) = Q(s, a) + \alpha * learnedvalue$

As with standard Q-Learning we update the Q-Value of the state-action with the results of the action, the impact of the change depending on the learning rate.

3.2.3 Differences

The comparison learner does not distinguish between a result that is a setback from the objective and a result that causes a failure state (other than a larger negative result heuristic). Our system does distinguish between these and it allows our system to make decisions that avoid absolute failure even if they result in temporary setbacks.

The comparison learner also takes into account state uncertainty in a more limited fashion than our system. It adjusts its learning rate by the number of times a state has been visited when determining how much a new action should effect the value of a state but it does not use uncertainty when determining which action to take.

Note that while our environment is set up so that rewards depend on where the agent ends up instead of the transitions it undergoes these two set ups are equivalent and our environment could be reformulated to be based on transitions without changing it. As such neither the comparison learner based on standard Q-Learning Watkins and Dayan (1992) nor the comparison learner based on Shen et al. (2014) are artificially disadvantaged by this set up.

3.3 Simplified examples

We will explain how these systems compare using a simple gridworld. In this gridworld there is a blue start location, a cyan current state and a green goal. Certain squares are risky and provide a chance of causing the system to enter a failure state - however the risk is not absolute and the system will enter a failure state less than 35% of the time it enters a dangerous square. Additionally even safe squares provide an extremely low chance of entering a failure state so a square cannot immediately be considered risky when a failure state is encountered.

White squares are estimated by the agent to be relatively safe with a low degree of uncertainty, red squares are estimated to be dangerous with a low degree of uncertainty, yellow squares are relatively unexplored and may be safe or dangerous, the risk of pink squares is uncertain but estimated to be high and the risk of purple squares is uncertain but estimated to be low. For simplicity we refer to gridsquares that have either never been visited or only been visited once

or twice and thus have a very high degree of uncertainty as 'unknown' gridsquares in this example.

Grey arrows represent the most likely action for each state according to the **danger training** mode policy, green arrows represent the most likely action for each state according to the **danger avoidance** mode policy and blue arrows represent most likely action for each state according to the policy of the comparison learner in the action selection phase with parameters set for exploitation. Multiple arrows of the same colour represent multiple options being roughly equally likely.

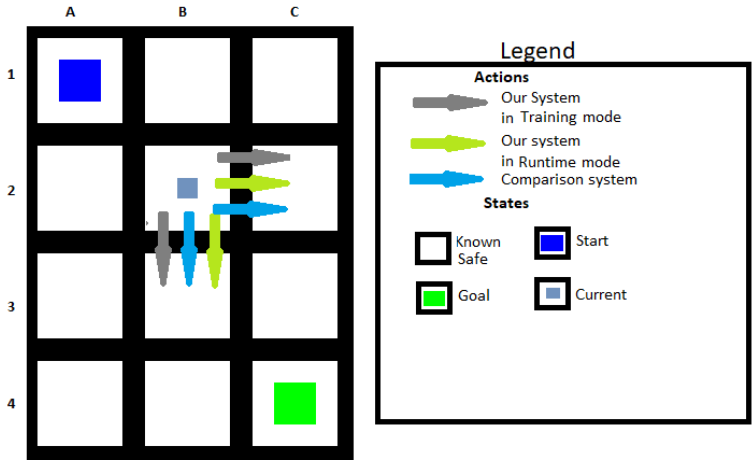

Figure 3.4: Example A. The simplest situation. There is a path to the goal and no risky squares are in the vicinity. All policies move towards the goal. The grey arrow represents our system in training mode (**danger training**) while the green arrow represents our system in **danger avoidance** mode.

In the example shown in Figure 3.4 all systems act similarly. Our system in **danger training** mode acts to maximise $Q(s,a)$ and $D(s,a)$ and minimise $C_Q(s,a))$ and $C_D(s,a))$; risk and uncertainty are uniformly low so it will maximise $Q(s,a)$ by being more likely to move towards the goal. It will likely enter either *B3* or *C2*. Our system in **danger avoidance** mode acts to maximise $Q(s,a)$,$C_Q(s,a))$ and $C_D(s,a))$ and minimise $D(s,a)$ - as risk and uncertainty are negligible the system maximises $Q(s,a)$, which is expressed in this scenario as proximity to the goal, by being more likely to move towards the goal. It will likely enter either *B3* or *C2*.

The comparison learner would assign the squares closer to the goal a higher Q-value and would be more likely to move towards them. It will likely enter either *B3* or *C2*.

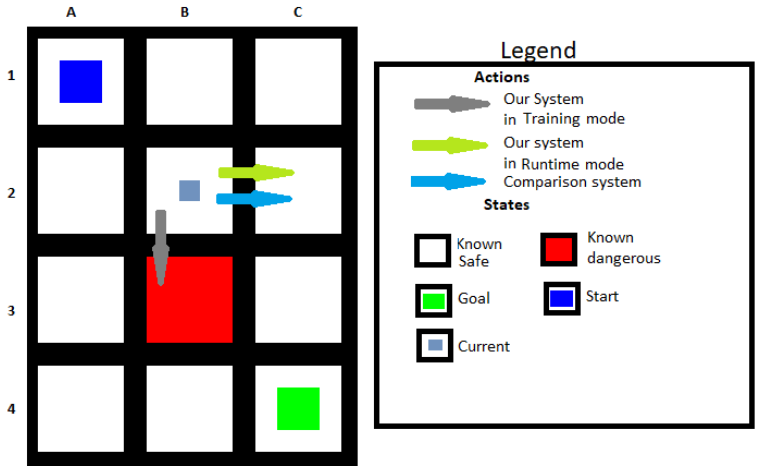

Figure 3.5: Example B. The simplest scenario that involves risk. Our system should seek the risk in training mode whereas other systems avoid it. The grey arrow represents our system in training mode (**danger training**) while the green arrow represents our system in **danger avoidance** mode.

The next example Figure 3.5 is similar to the previous example but with a known risky square placed between the current state and the goal.

In **danger training** mode the system attempts to maximise $Q(s, a)$ and $D(s, a)$ but with a bias in favour of $D(s, a)$. As such is most likely to enter the risky square - it will likely move to *B3*.

Note that future work involves alternating between **danger training** mode (which learns $D(s, a)$ more effectively) and **danger avoidance** mode (which learns $Q(s, a)$ more effectively) during training to ensure that the system learns both $Q(s, a)$ and $D(s, a)$.

In **danger avoidance** mode the system attempts to maximise $Q(s, a)$ by minimising the distance to the goal and minimise risk as represented by $D(s, a)$ and so is likely to move to the right into *C2* - moving closer to the goal while avoiding risk.

The comparison learner is likely to have a slight preference for moving to the right into *C2* since both answers move close to the goal but the risky action sometimes fails and gives no reward.

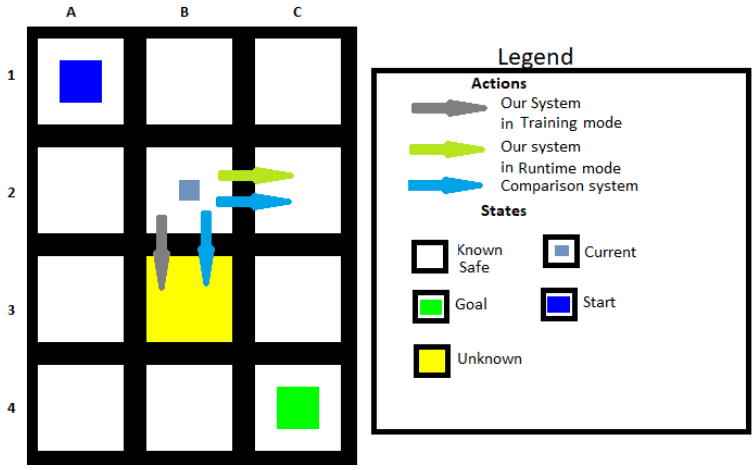

Figure 3.6: Example C. The unexplored square scenario. Our system will seek the uncertainty in training mode and avoid it in runtime mode. The comparison learner will not detect it and will just move towards the goal. The grey arrow represents our system in training mode (**danger training**) while the green arrow represents our system in **danger avoidance** mode.

In Figure 3.6 there is a relatively unexplored state between the current state and the goal, in *B3*. Though, absent application specific special circumstances, it is unlikely for the system to encounter an uncertain state surrounded by known states this example is presented to show how the system handles unknown states in general.

In **danger training** mode our system will attempt to maximise $Q(s, a)$ and minimise certainty as represented by $C_Q(s, a))$ and $C_D(s, a))$ and so is likely to move into the uncertain square - it will likely move into *B3*. In **danger avoidance** mode the system will attempt to maximise $Q(s, a)$ by minimising the distance to the goal and minimising uncertainty and is likely to pick the safe option of moving to the right which moves it closer to the goal without risk. It will likely move into *C2*.

The comparison learner's response will vary greatly depending on whether a failure state was encountered in the one or two times the square was visited. If it has encountered a failure state the system will treat the state as a highly negative reward with a very low q-value and move to the right. If the system did not encounter a failure state the system will treat the state as safe and will be just as likely to move into the uncertain square as it is to move to the right. If it has not been visited at all it will have the system's default Q-value which will likely make it less likely to be visited than the known safe space but not as much

as if it had encountered a failure state. It will likely move into either *B3* or *C2*.

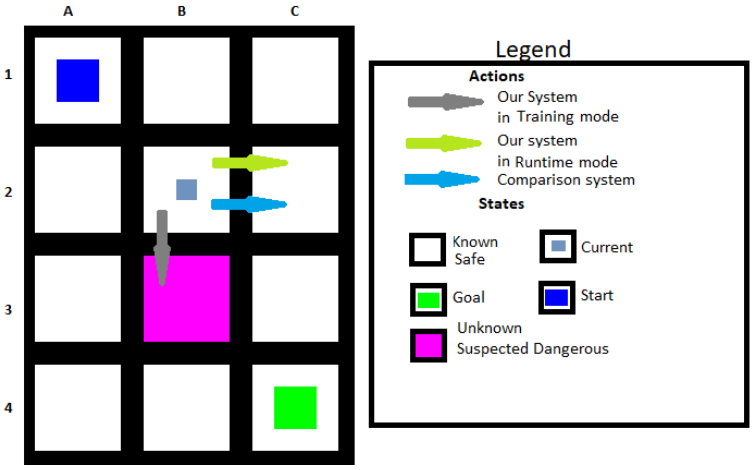

Figure 3.7: Example D1. Our system will seek the danger and uncertainty in training mode. Both our system in runtime mode and the comparison learner will avoid it, though our system will be less likely to pick it. The grey arrow represents our system in training mode (**danger training**) while the green arrow represents our system in **danger avoidance** mode.

In Figure 3.7 there is a relatively unexplored square, suspected to be dangerous in B3, between the current state and the goal.

In **danger training** mode our system will attempt to maximise $Q(s, a)$ and $D(s, a)$ and minimise $C_Q(s, a))$ and $C_D(s, a)$. As such the agent will move to attempt to confirm the state is dangerous, it will likely move to *B3*. In **danger avoidance** mode our system will attempt to maximise $Q(s, a)$, $C_Q(s, a))$ and $C_D(s, a))$ and minimise $D(s, a)$. It will, as such, avoid the potentially risky square. It will likely move to *C2*.

As stated before if the comparison learner has encountered a couple of failures before it will rate the action as being less advantageous and will be more likely to pick another option. It will likely move to *C2*.

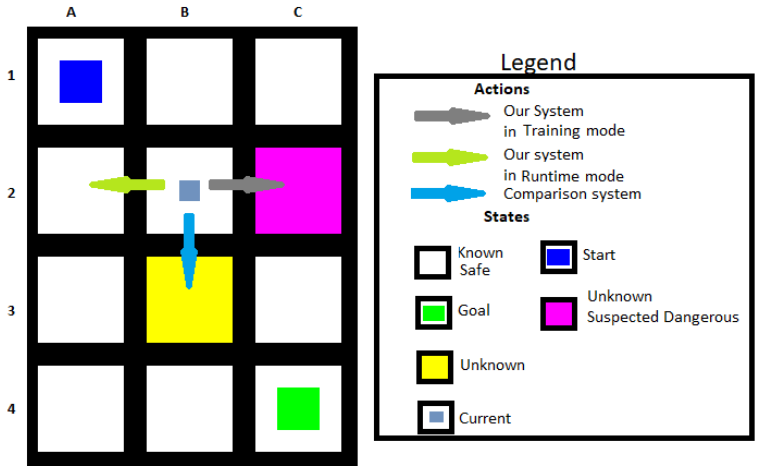

Figure 3.8: Example D2. A choice between an uncertain square and a suspected dangerous state. Our system will seek the danger in training mode, move around in runtime mode whereas the comparison learner will not take uncertainty into account and go through it. The grey arrow represents our system in training mode (**danger training**) while the green arrow represents our system in **danger avoidance** mode.

In Figure 3.8 the system is present with a choice between moving into a unexplored state not suspected to be safe or dangerous at *B3*, a relatively unexplored state suspected to be dangerous at *C2* and a longer path around in order to reach the goal, going from *B2* through *A2*-*A3*-*A4*-*B4* and finally to *C4*. There is some chance of the system returning to *B2* when it reaches *A2* but as actions are determined probabilistically it will not get stuck in an infinite loop and will likely move to A3 soon.

In **danger training** mode the system will attempt maximise $Q(s, a)$, $D(s, a)$ and minimise $C_Q(s, a))$ and $C_D(s, a))$. This will cause it to prioritise the square suspected to be dangerous (low $C_Q(s, a)$, low $C_D(s, a)$ and high $D(s, a))$ over the uncertain state not suspected to be safe or dangerous (low $C_Q(s, a)$ and low $C_D(s, a))$. It will likely move to *C2*. In **danger avoidance** mode the system will attempt to maximise $Q(s, a)$, $C_Q(s, a))$ and $C_D(s, a))$ while minimising $D(s, a)$. The system will be more likely to traverse the uncertain state than the suspected risky state but will be even more likely to move around both. It will likely move to *A2*.

If the state has not been visited at all comparison learner's action depends on the default Q-Value. If the state is relatively uncertain but has been visited at least once in a run that reached the goal the comparison learner will likely

choose to attempt the uncertain state, having a higher Q-value than either the risky state or moving away from the goal. It will likely move to *B3*.

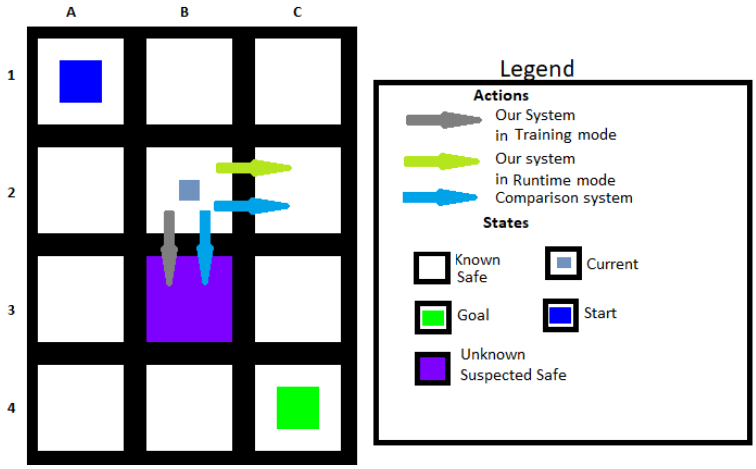

Figure 3.9: Example E1. The suspected safe space scenario. Our system will seek the danger in training mode, avoid it in runtime mode whereas the comparison learner will be unable to distinguish between the two states and will cost one of them at random. The grey arrow represents our system in training mode (**danger training**) while the green arrow represents our system in **danger avoidance** mode.

In Figure 3.9 the system is presented with a state that is relatively unexplored but suspected to be safe in B3.

In **danger training** mode our system will attempt to maximize $Q(s, a)$ and $D(s, a)$ and minimise $C_Q(s, a))$ and $C_D(s, a))$. It will likely move into the uncertain space. It will likely move to *B3*. In **danger avoidance** mode our system will attempt to maximise $Q(s, a)$, $C_Q(s, a))$ and $C_D(s, a))$ and minimise $D(s, a)$ by choosing the option that brings it equally close to the goal while avoiding uncertainty and moving to the right. It will likely move to *C2*.

Our comparison learner will be unlikely to be able to distinguish between the two states and will be equally likely to pick either of them. It will likely pick either *B3* or *C2*.

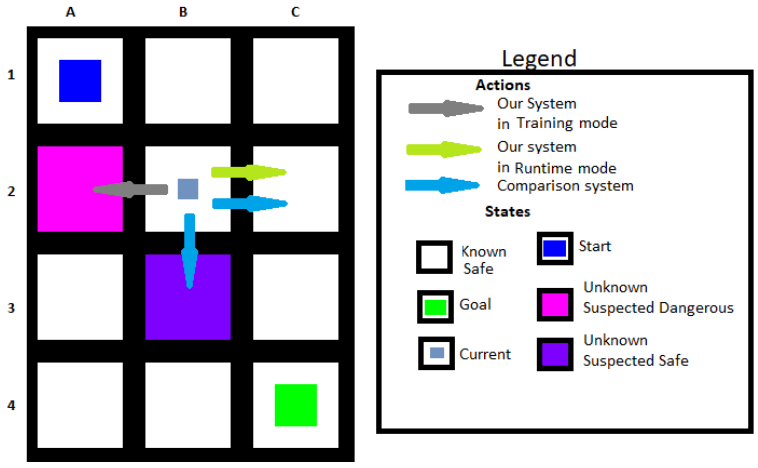

Figure 3.10: Example E2. A choice between suspected safety and suspected danger. In training mode our system will prioritise exploring the dangerous state over moving to the goal and will move to it, in runtime mode our system will move to avoid both obstacles and the comparison learner will be unable to distinguish between the suspected safe state and a confirmed safe state. The grey arrow represents our system in training mode (**danger training**) while the green arrow represents our system in **danger avoidance** mode.

In Figure 3.10 the system is presented with a choice between an uncertain state suspected to be dangerous further from the goal in *A2* and an uncertain state suspected to be safe close to the goal in *B3*.

The system in **danger training** mode will attempt to maximise $Q(s, a)$ and $D(s, a)$ and minimise $C_Q(s, a))$ and $C_D(s, a))$. However when forced to choose between exploring states with higher $Q(s, a)$ and states with higher $D(s, a)$ it will be more likely to explore the risky states and as such will likely move to the left. It will likely move to *A2*.

The system in **danger avoidance** mode will act as it did in Figure 3.9. It will choose the safe option of moving to the right, it will likely move to *C2*.

Our comparison learner will likewise act as it did in Figure 3.9. It will be unable to distinguish between moving down and to the right. It will either move to *B3* or *C2*.

Having shown how our system operates in theory we will now show how the novel system we have described performs relative to the comparison learners.

Chapter 4

Separate Reward and Risk Metrics

The first novel contribution is concerned with the fashion in which our system models risk and its implications on exploration during training. Our system considers risk not as a negative reward like traditional risk sensitive reinforcement learning but as a separate quantity that the system learns independently of reward.

This more accurately represents many use cases such as a robot navigating a potentially dangerous environment - a failure state may not merely represent not achieving the immediate objective but may involve damaging or destroying the robot, which has longer term implications.

More importantly the system is able to consider both of these qualities separately during both the training and exploitation phases.

In **danger training** mode the learner is able to maximise both eventual reward and risk. This ensures that not only are potential paths to the goal explored but dangerous regions are also mapped out in detail. This means our system is able to avoid the problem described in Sammut et al. (1992) where dangerous areas are avoided in exploration to the extent that these dangerous areas remain relatively unexplored. For instance a human driver won't choose to drive a robot on the edge of a cliff. This can be acceptable in some scenarios given that the agent will never choose to enter these regions. However if there is a chance of the system encountering these states regardless, as will sometimes occur in the real world, the agent will be unable to make sensible decisions if the agent does enter these regions.

In **danger avoidance** mode the learner is still able to consider eventual

reward and risk separately. Whereas a traditional risk sensitive learner may view a risky action that moves the system closer to the goal by a considerable distance equivalent to a safe action that moves the system slightly away from the goal, our system is able to distinguish between them and select the safer option.

Both **danger training** and **danger avoidance** may be used during training but only **danger avoidance** is used during exploitation.

If the agent transitions to a failure state our system begins a new run without making any further actions and the current run is not considered a success. A success is defined as a run in which the agent successfully reaches the goal square without transitioning to a failure state.

4.1 Effect of separate risk metric on Q-Learning

Set up

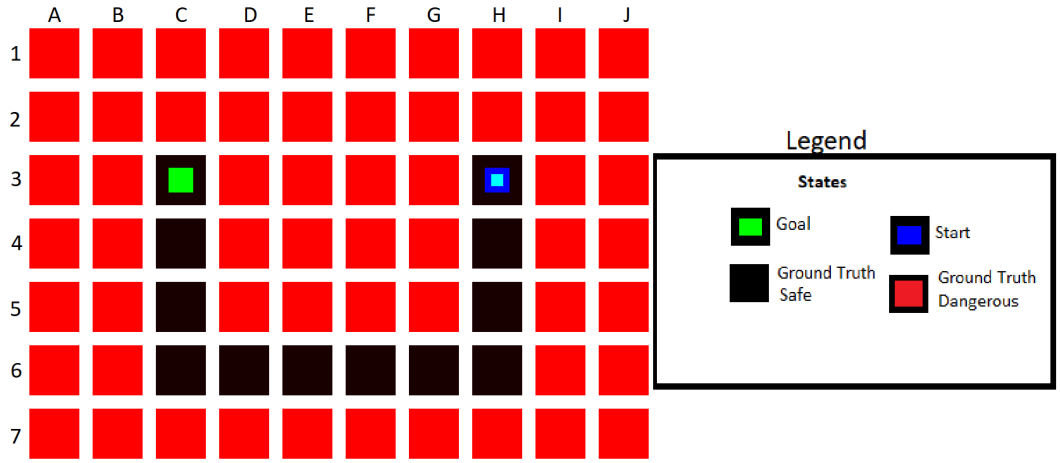

Figure 4.1: The simple safe path example world. The system must follow the u curve in order to remain safe while progressing to the goal.

The runs were performed on the simple safe path world shown in Figure 4.1 configured so that the safe gridsquares have a 0% chance of causing the agent to enter a failure state, the dangerous gridsquares have a 35% chance of causing the agent to enter a failure state and all $P_M(s, a)$ values are set to 0.9. Runs were performed using 3 different set ups: standard Q-Learning, our learner in secondary learner mode using both **danger training** mode and **danger avoidance** mode during training and our learner using only **danger avoidance** mode during training.

Aim

We aim to show how our learner performs against standard Q-Learning given ample training examples for both learners and also show the effect of **danger training** mode with many training data examples.

Data Generation

Each of the 3 learner set ups is allowed to run for a million steps in each section. Between the first and second section the mixed **danger training** mode/**danger avoidance** mode example will switch from **danger training** mode to **danger avoidance** mode. The proportion of runs that succeed in a 20 run moving window was determined and then a median filter over 100 runs applied to this success proportion. This was then used to determine how the success rate changes over time.

Results

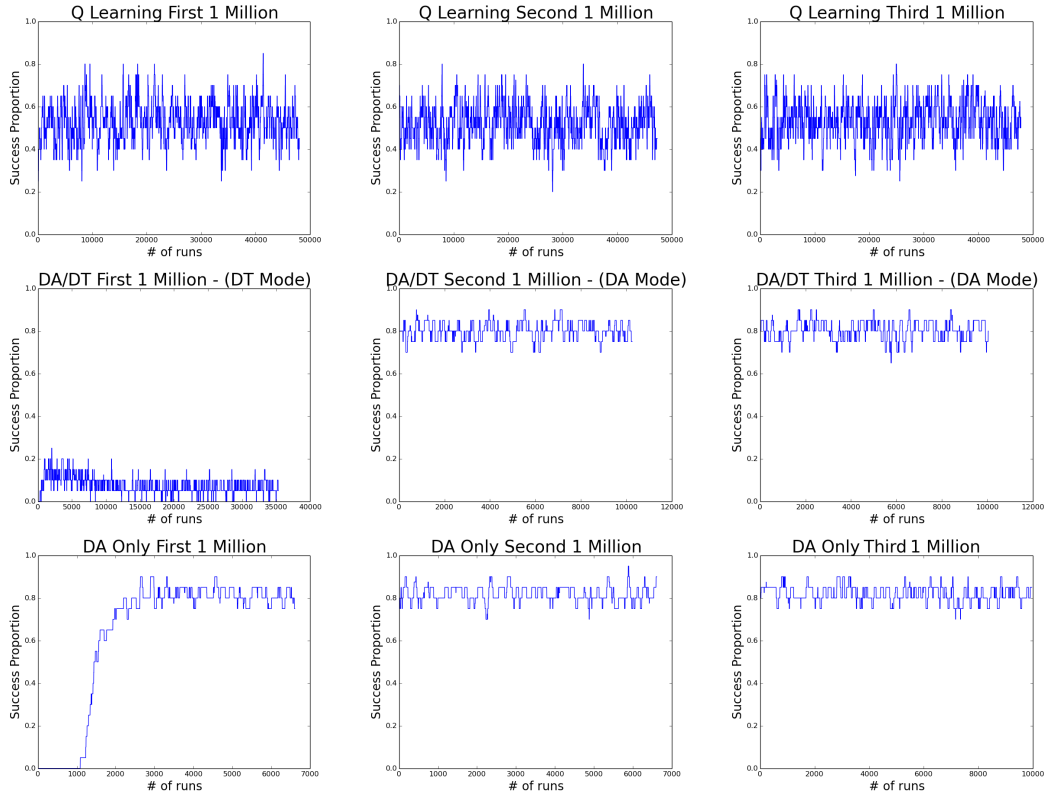

Figure 4.2: The proportion of runs that succeed in a 20 run moving window which then has a median filter applied to it. Each graph represents a million steps with each experimental run composed of three million step stages, two in training and the third in exploitation. The first row is standard Q-Learning. The second row is our learner using both **danger training** and **danger avoidance** mode, switching between stage 1 and 2. The third row is our system using **danger avoidance** mode only. Note that though each graph represents the same number of steps the number of runs will differ from graph to graph as not all runs are composed of the same number of steps.

The results for this experiment are shown in Figure 4.2. These graphs represent the proportion of runs that succeed in a 20 run moving window. Every row represents a different experimental set up; the first row uses standard Q-Learning, the second row uses our learner configured so that it uses **danger training** mode followed by **danger avoidance** mode and the third row uses our learner configured to use **danger avoidance** mode only. Each column represents a stage of the experiment, each composed of one million steps. The first two columns are training, with the learner using both **danger training** and **danger avoidance** mode switching from the former to the latter between the first and second stage. The last column is exploitation and represents how that learner performs once trained.

Each graph is consistent in the number of steps it represents but the x-axis, the number of runs, will differ from graph to graph as the number of steps each run is composed of will differ depending on how quickly the agent finds the goal or encounters a failure state. The number of runs is used as the x-axis as, though each example is composed of a set number of steps, success/failure is a property of runs. Notably runs that encounter a failure state early take fewer steps than any run that reaches the goal so stages with a lower success rate will tend to have more runs than stages with a higher success rate.

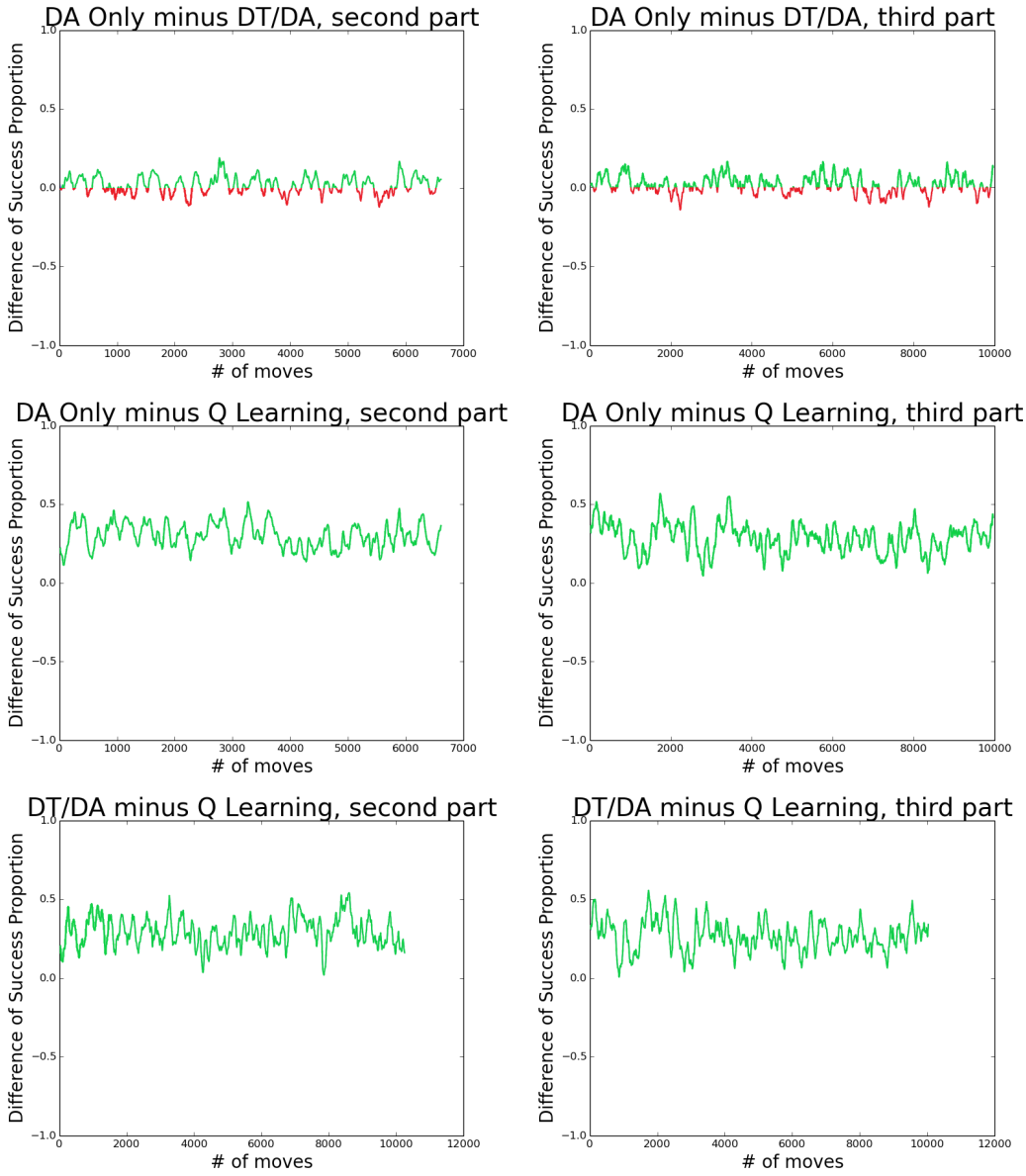

Figure 4.3: This figure shows the differences between the performance of different learner set ups in the second and third stages in Figure 4.2. The first column shows the differences of the second stage and the second column shows the differences of the third stage. The first row shows the difference between our learner using only **danger avoidance** mode and our learner using both **danger training** mode and **danger avoidance** mode. The second row shows the difference between our learner using only **danger avoidance** mode and standard Q Learning. The third row shows the difference between our learner using both **danger training** mode and **danger avoidance** mode and standard Q Learning.

As shown in the differences in Figure 4.3, our learner in secondary learner mode outperforms standard Q-Learning during the exploitation section in both the mixed **danger training** Mode/**danger avoidance** mode example and the **danger avoidance** mode only example. This indicates that in this environment our learner correctly identifies the safe path and traverses it more often than standard Q-Learning which often cuts straight across the dangerous dangerous squares to reach the goal unnecessarily.

Figure 4.3 also shows that our learner using the mixed **danger training** mode/**danger avoidance** mode training experimental set up does not outperform the **danger avoidance** mode only experimental set up. This suggests that eventually given enough runs a policy generated using only **danger avoidance** mode will be equivalent to a policy generated using a mixture of **danger training** mode and **danger avoidance** mode. Given a fixed starting and goal location the agent encounters all the squares adjacent to the safe path enough times to generate accurate $Q(s, a)$ and $D(s, a)$ values despite not actively exploring.

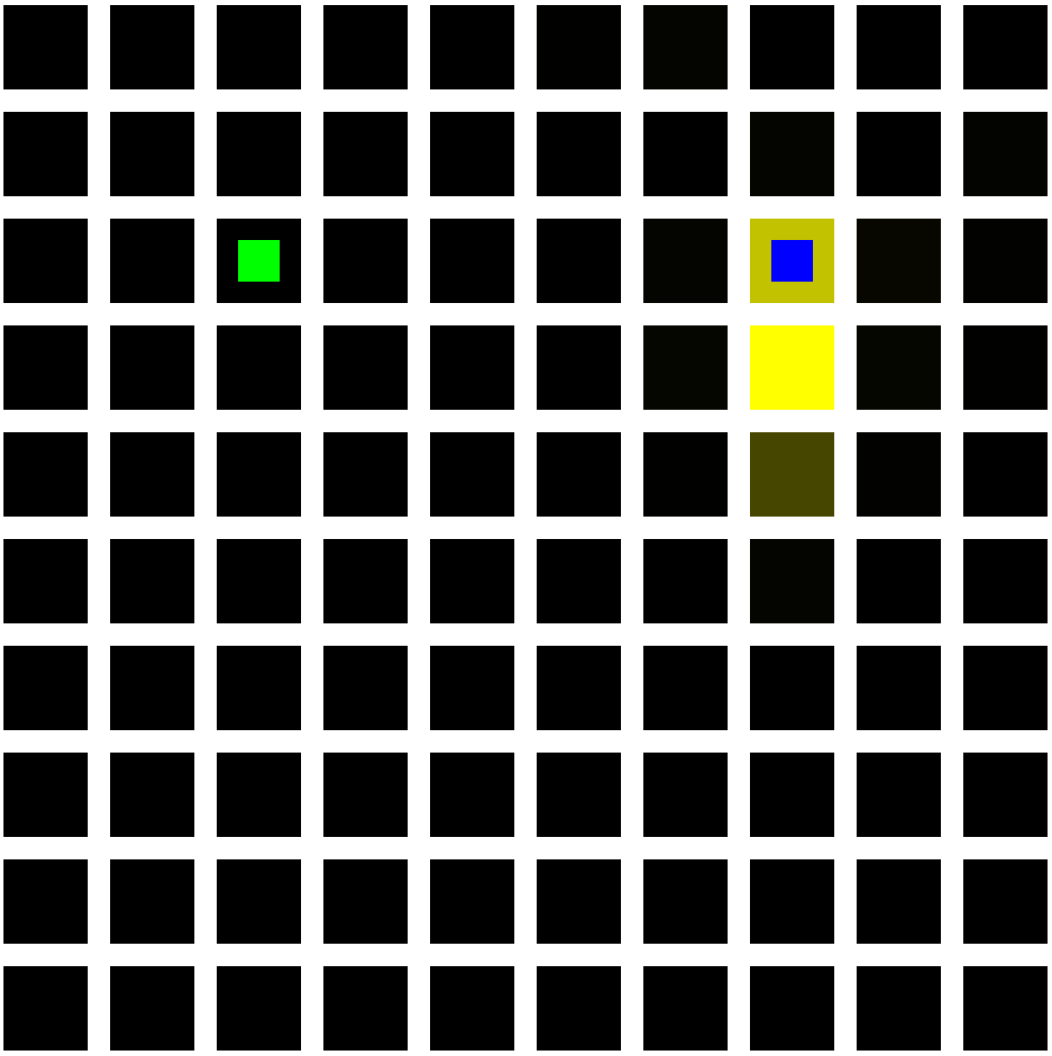

Figure 4.4: A heatmap of the number of times the learner visited particular states during the first 10,000 steps in **danger avoidance** mode only.

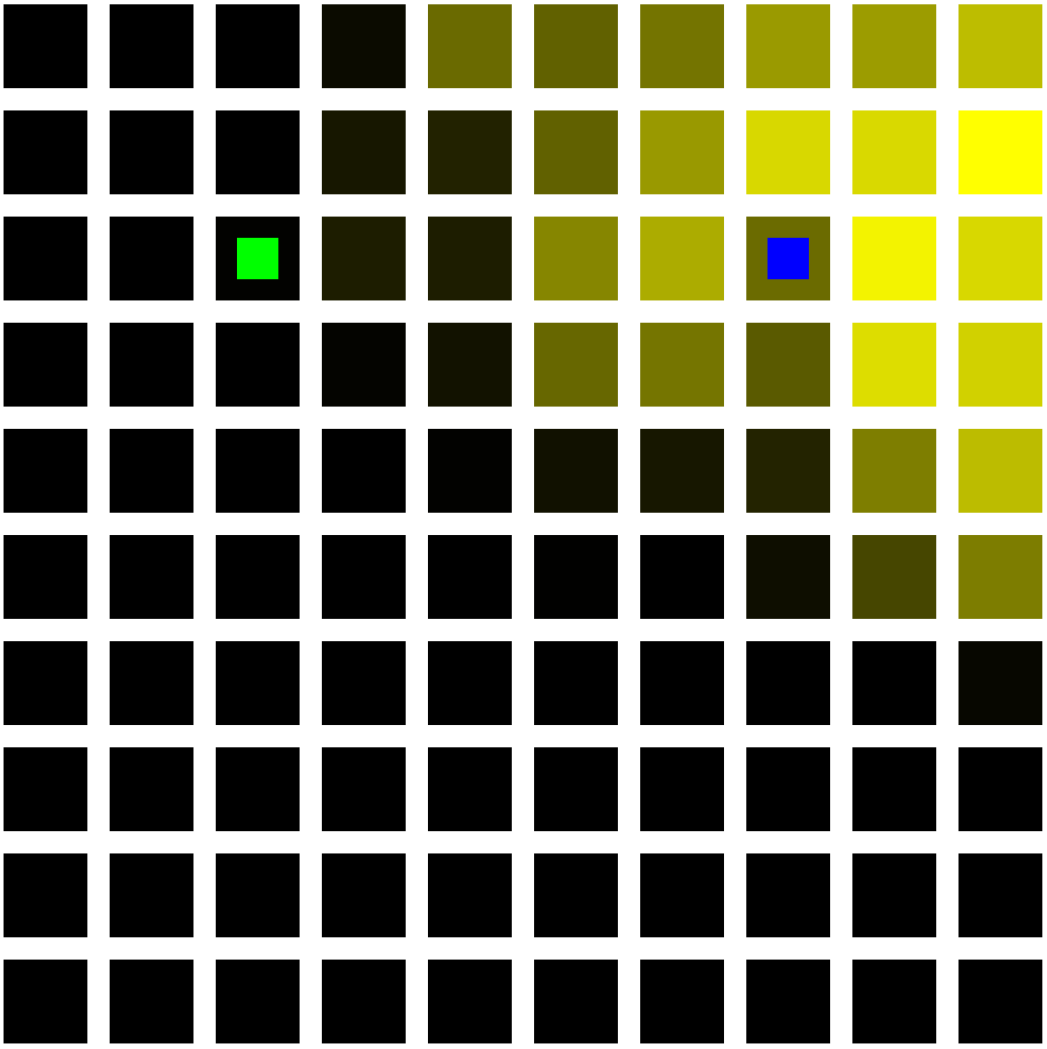

Figure 4.5: A heatmap of the number of times the learner visited particular states during the first 10,000 steps in **danger training** mode.

However at first our learner using the **danger avoidance** mode only experimental set up achieves no successes. The learner has not encountered the goal and does not have any information about where it is located but instead of exploring the environment it chooses to revisit the safe states located right next to the start location. This is shown in Figure 4.4 which is a heatmap of the first 10,000 steps of a **danger avoidance** mode experimental set up, the agent does not explore the environment as a whole but keeps revisiting the states it knows to be safe. Compare this to Figure 4.5 in which the agent explores its immediate surroundings during the first 10,000 steps in **danger training** mode, the system explores the environment as a whole with a preference for states off of the safe

path due to the learner seeking danger in this mode. This suggests that a shorter run of **danger training** mode may aid in initial exploration. Once the agent finds the goal, performance quickly improves as the agent begins following the safe path.

4.2 Danger Training Duration Experiment

Set up

As in the prior example in Section 4.1 this experiment will be run in the Simple Safe Path world but with the $P_D(s)$ value of the safe squares set to 0, the $P_D(s)$ value of the dangerous squares set to 0.35 and all $P_M(s, a)$ values set to 0.9. The number of training steps will be varied in this experiment, each simulation of a particular number of training steps will be performed using our learner in secondary learner mode, each simulation transitioning from **danger training** mode to **danger avoidance** mode after the training runs. Additionally standard Q-Learning runs of the same lengths will be performed as a comparison.

Aim

To determine the effect of varying the length of training in **danger training** mode has on learner performance and the ideal transition point between **danger training** mode and **danger avoidance** mode.

Data Generation

Our learner will run for a varying number of steps in **danger training** mode. Then the learner will run for a further 50,000 runs in **danger avoidance** mode. The same number of runs for both will be performed by standard Q-Learning as a comparison. The second set of runs will not have their parameters set to exploitation for either learner to mirror the second column of results shown in Figure 4.2. These results still represent an approximation of how each learner would function during exploitation given the similarities between the second and third columns of results for all examples. The success rate in a 20 run moving window will be displayed.

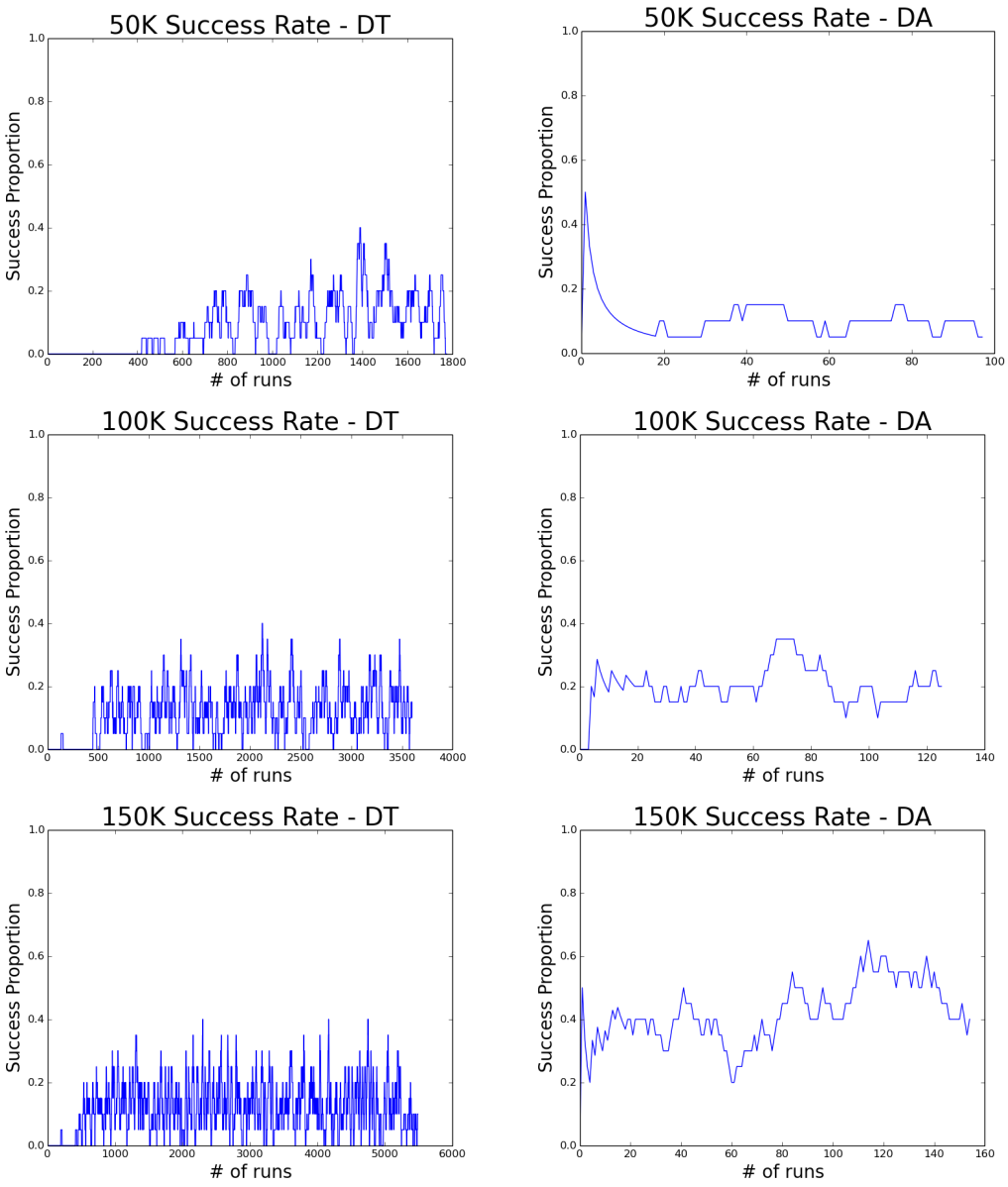

Figure 4.6: (1/2) The proportion of runs that succeed in a 20 run moving window. Each graph on the left hand side is that many steps in **danger training** mode. Each graph on the right hand side is 50K runs in **danger avoidance** mode after being trained by the right hand runs. Note that comparison should be performed vertically with the evaluation of the effectiveness of the training on the right side rather than between the right and left boxes. A box plot of these results is shown in Figure 4.10.

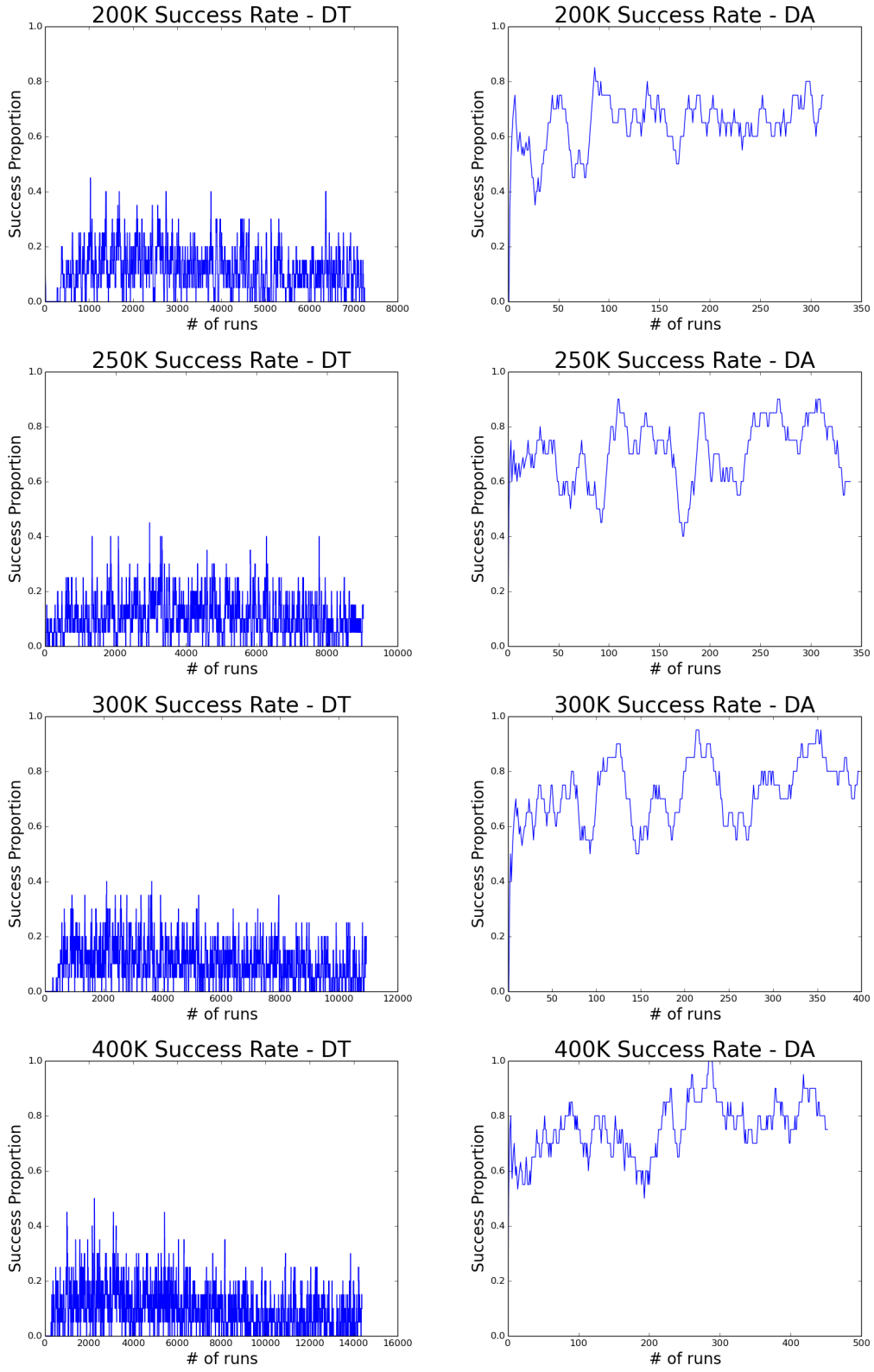
200K Success Rate - DT
200K Success Rate - DA
250K Success Rate - DT
250K Success Rate - DA
300K Success Rate - DT
300K Success Rate - DA
400K Success Rate - DT
400K Success Rate - DA
Success Proportion
# of runs

Figure 4.7: (2/2)

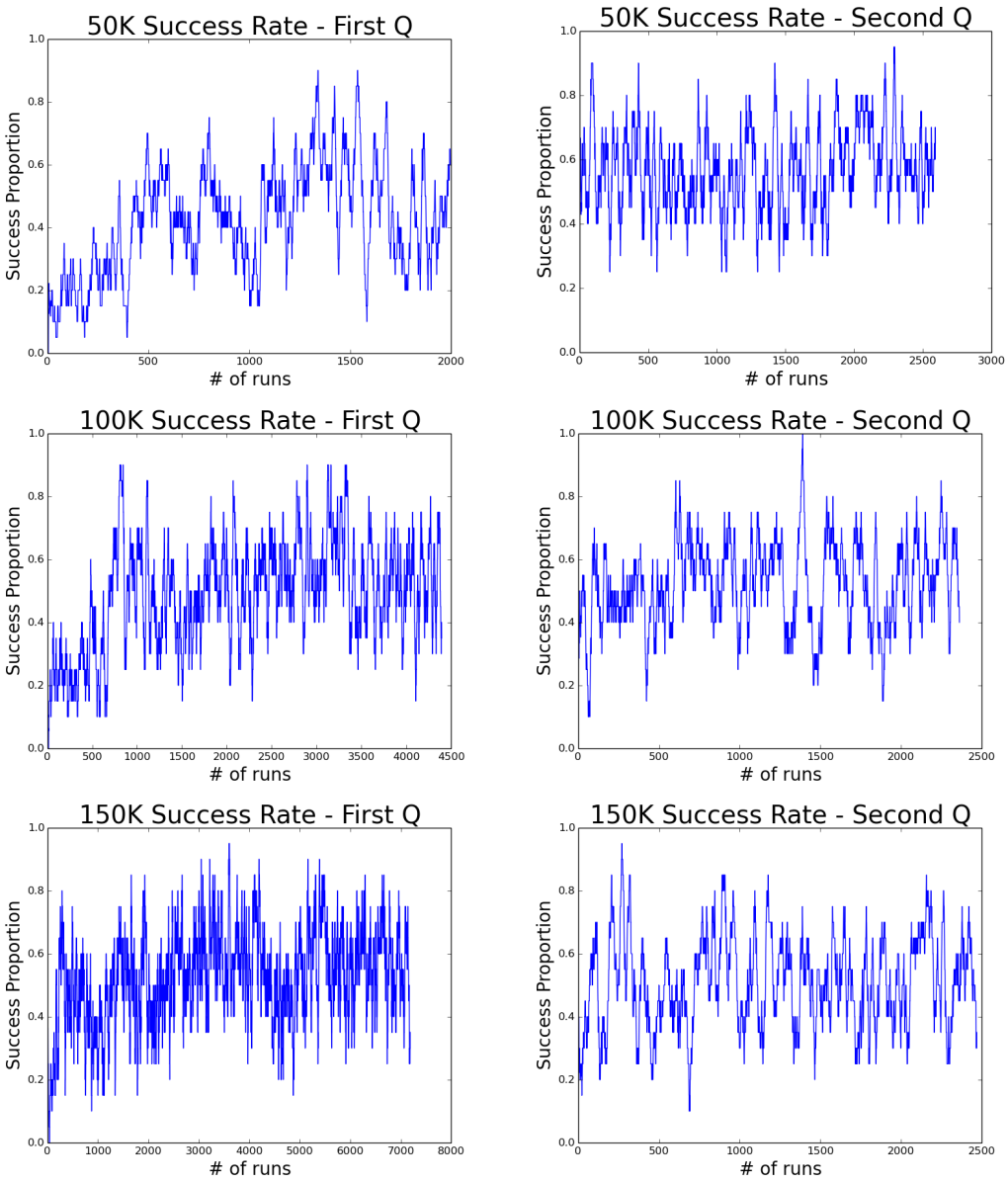

Figure 4.8: (1/2) The proportion of runs that succeed in a 20 run moving window. Each graph on the left hand side is that many steps training Q-Learning. Each graph on the right hand side is a further 50K runs exploiting Q-Learning. Note that comparison should be performed vertically with the evaluation of the effectiveness of the training on the right side rather than between the right and left boxes. A box plot of these results is shown in Figure 4.10.

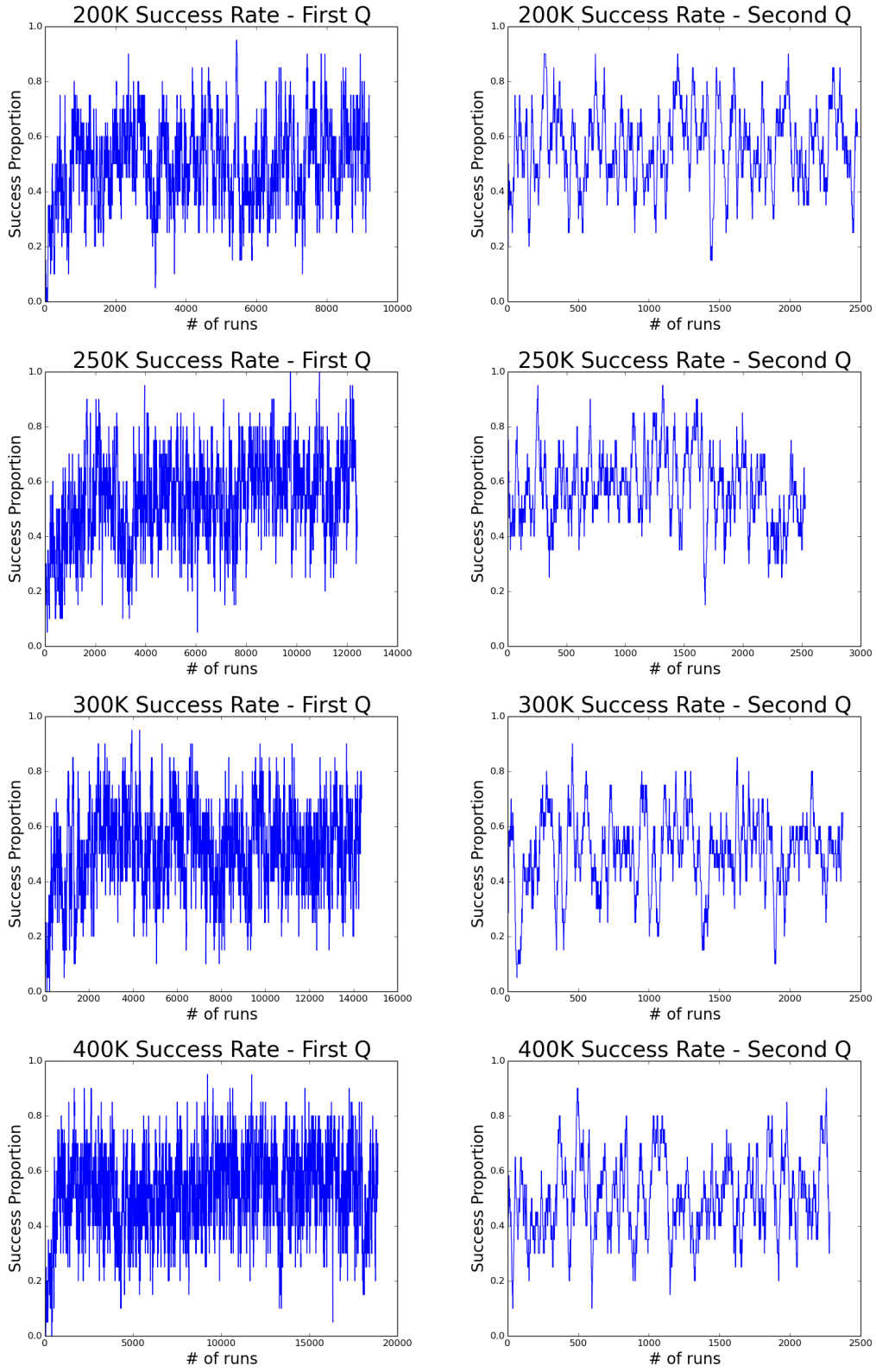
200K Success Rate - First Q
200K Success Rate - Second Q
250K Success Rate - First Q
250K Success Rate - Second Q
300K Success Rate - First Q
300K Success Rate - Second Q
400K Success Rate - First Q
400K Success Rate - Second Q
Success Proportion
# of runs

Figure 4.9: (2/2)

The results for our learner in this experiment are shown in Figures 4.6 and 4.7, a comparison using standard Q-Learning is shown in Figures 4.8 and 4.9. As in the prior experiment note that since each example is a pre-determined number of steps but success/failure is a properly of runs and each run can have an different number of steps the length of each plot may not be consistent with the others in runs but it is the pre-determined number of steps

These results show that our learner does explore in **danger training** mode, eventually to the point of skipping the initial lack of successes when the learner starts in **danger avoidance** mode.

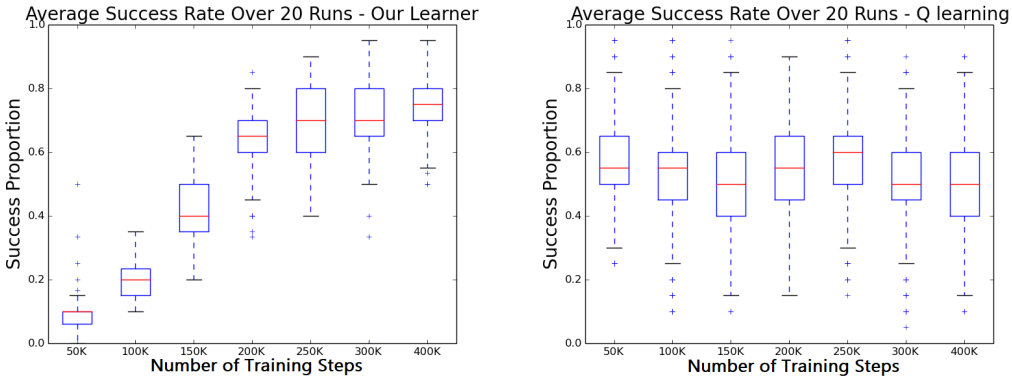

Figure 4.10: Box plots of the results shown in Figures 4.6 and 4.7 (left) and Figures 4.8 and 4.9 (right). These plot represent the average success proportion over 20 runs in the second 50,000 step stage after each learner had been trained for a varying number of steps.

Boxplots of these results are shown in Figure 4.10. These show that after 250K steps training in **danger training** mode the learner converges on its policy and further training runs show diminishing returns. As such this the ideal transition point from **danger training** mode to **danger avoidance** mode in most applications. However if minimum or maximum values are important (for instance an application where avoiding a string of failures is important beyond reducing the average failure rate) the ideal transition point may be after 300K steps or more.

Additionally after 200K steps training in **danger avoidance** mode our learner begins consistently outperforming standard Q-Learning. Standard Q-Learning converges much more quickly than our learner, starting with a much higher success rate than our learner but not showing any signs of improvement given further training runs. This is partially explained by the Q-Learner only needing to learn

Q-Values and not $D(s, a)$ values. However its average post-convergence success rate is significantly lower than our learner's average post-convergence success rate. This is because our learner is purposefully designed to detect which actions are more dangerous and avoid them whereas standard Q learning does not take this explicitly into account and will take a much longer time to converge to an optimal solution that takes these dangers into account.

4.3 Effect of Random Variation on Results

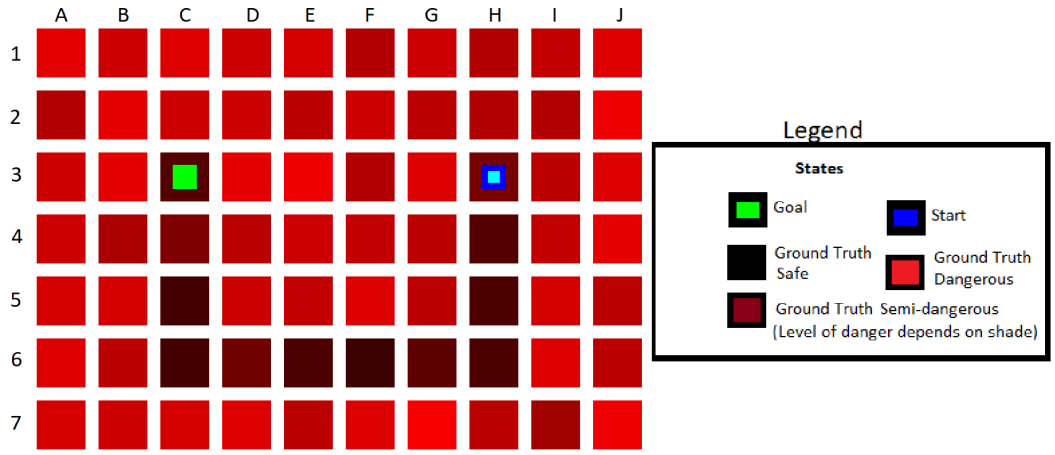

Figure 4.11: The random safe path example world. As with the last example but with added randomness to make learning more difficult.

Set up

The runs were performed on the Random Safe Path world shown in Figure 4.11. Three variations of this world were used: one in which the safe path still has no chance of causing the agent to transition to a failure state ($P_D(s) = 0$), one in which the safe path has a probability of causing the agent to transition to the failure state that is always lower than the dangerous regions but still relatively high ($0.03 \leq P_D(s) \leq 0.15$) and one in which the safe path has a non-zero but very low chance of causing the agent to transition to a failure state ($0 \leq P_D(s) \leq 0.03$). As in Section 4.1 all P_M values were set to 0.9.

Aim

We aim to show how our learner performs against standard Q Learning given ample training examples for both learners in environments with different levels of danger randomness, determining the situations in which our learner continues to outperform standard Q-Learning.

Data Generation

The set up was the same as in Section 4.1 but with the Random Safe Path worlds. Each of the examples is allowed to run for a million steps in each section. Between the first and second section the mixed **danger training** mode/**danger avoidance** mode example switched from **danger training** mode to **danger avoidance** mode. The proportion of runs that succeed in a 20 run moving window was determined and then a median filter over 100 runs applied to this success proportion. This was then used to determine how the success rate changes over time.

Results

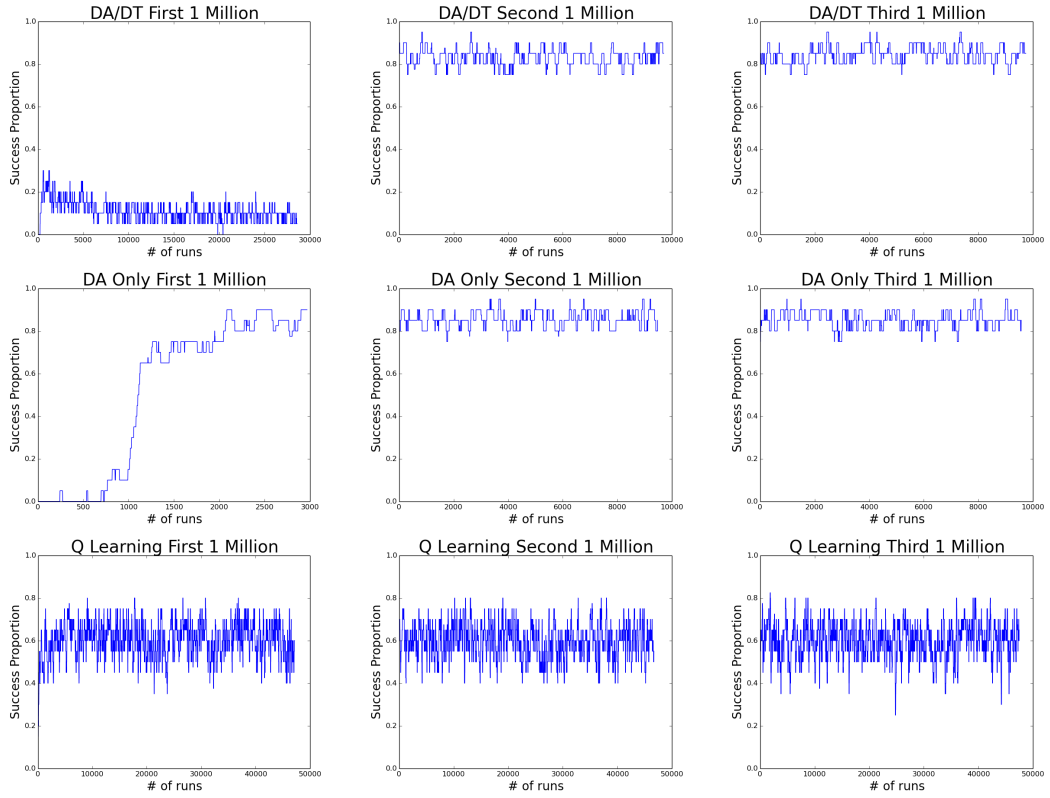

Figure 4.12: The proportion of runs that succeed with the danger values of the safe path set to zero in a 20 run moving window which then has a median filter applied to it. Each graph represents a million steps with each experimental run composed of three million step stages, two in training and the third in exploitation. The first row is our learner using both **danger training** and **danger avoidance** mode, switching between stage 1 and 2. The second row is our system using **danger avoidance** mode only. The third row is standard Q-Learning. Note that though each graph represents the same number of steps the number of runs will differ from graph to graph as not all runs are composed of the same number of steps.

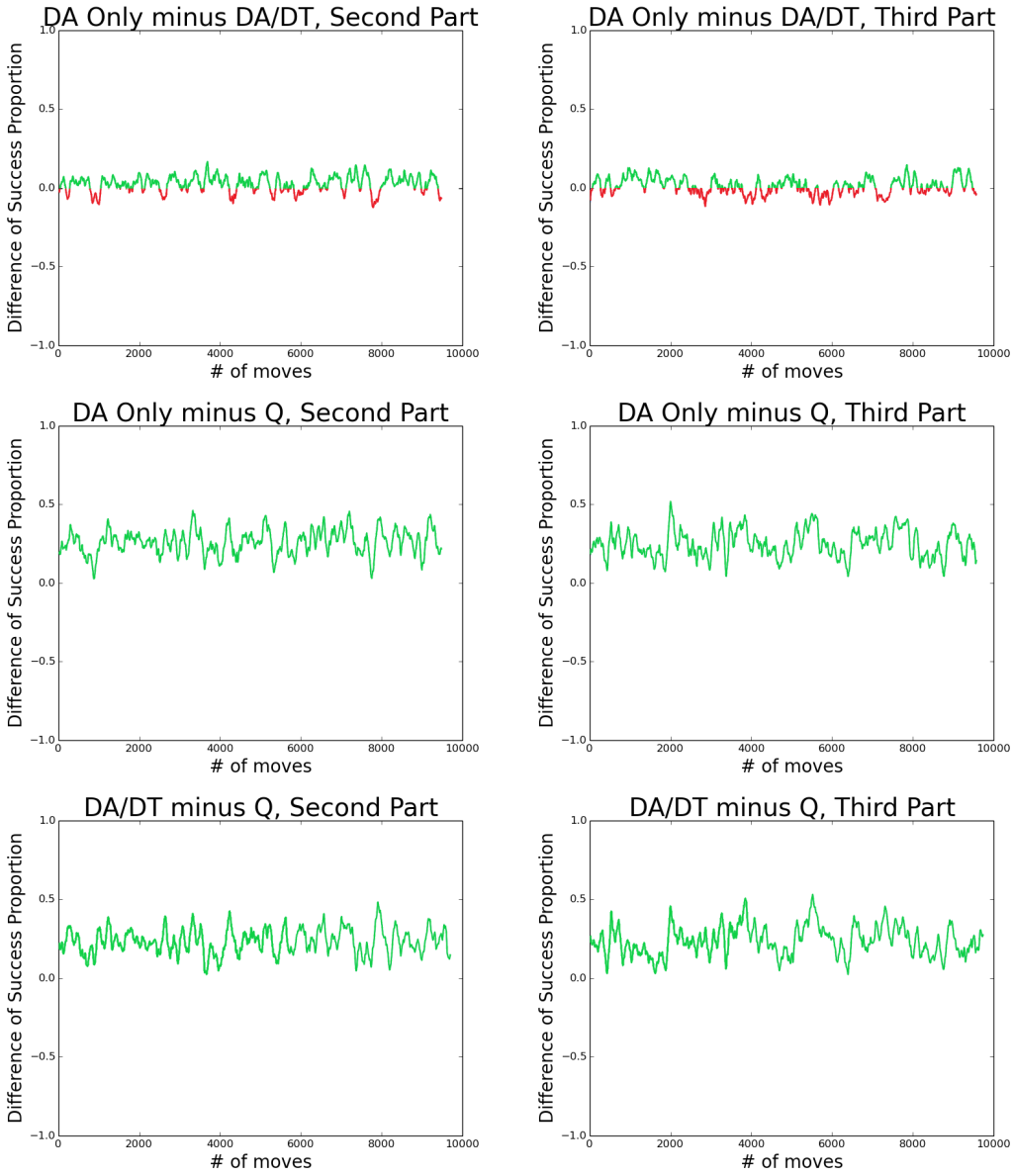

Figure 4.13: This figure shows the differences between the performance of different learner set ups in the second and third stages in Figure 4.12. The first column shows the differences of the second stage and the second column shows the differences of the third stage. The first row shows the difference between our learner using only **danger avoidance** mode and our learner using both **danger training** mode and **danger avoidance** mode. The second row shows the difference between our learner using only **danger avoidance** mode and standard Q Learning. The third row shows the difference between our learner using both **danger training** mode and **danger avoidance** mode and standard Q Learning.

The results for the experimental run with the $P_D(s)$ values on the safe path set to zero are shown in Figure 4.12. Additionally the differences between the performance of each experimental set up are shown in Figure 4.13.

In this experiment the random danger variation in the dangerous region does not substantially alter the results found in Section 4.1. The system still successfully finds and follows the safe path to the goal.

Next we will examine the impact of introducing random danger variation to the safe path.

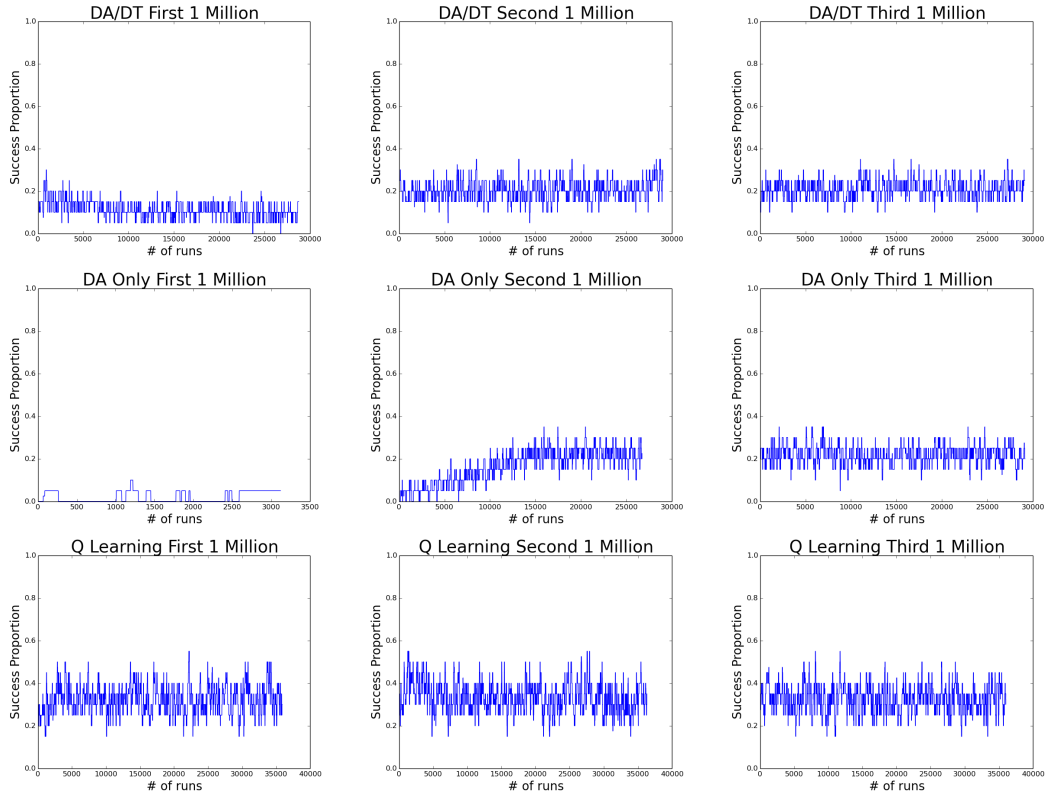

Figure 4.14: The proportion of runs that succeed with danger values on the safe path set to a random value in a 20 run moving window which then has a median filter applied to it. Each graph represents a million steps with each experimental run composed of three million step stages, two in training and the third in exploitation. The first row is our learner using both **danger training** and **danger avoidance** mode, switching between stage 1 and 2. The second row is our system using **danger avoidance** mode only. The third row is standard Q-Learning. Note that though each graph represents the same number of steps the number of runs will differ from graph to graph as not all runs are composed of the same number of steps.

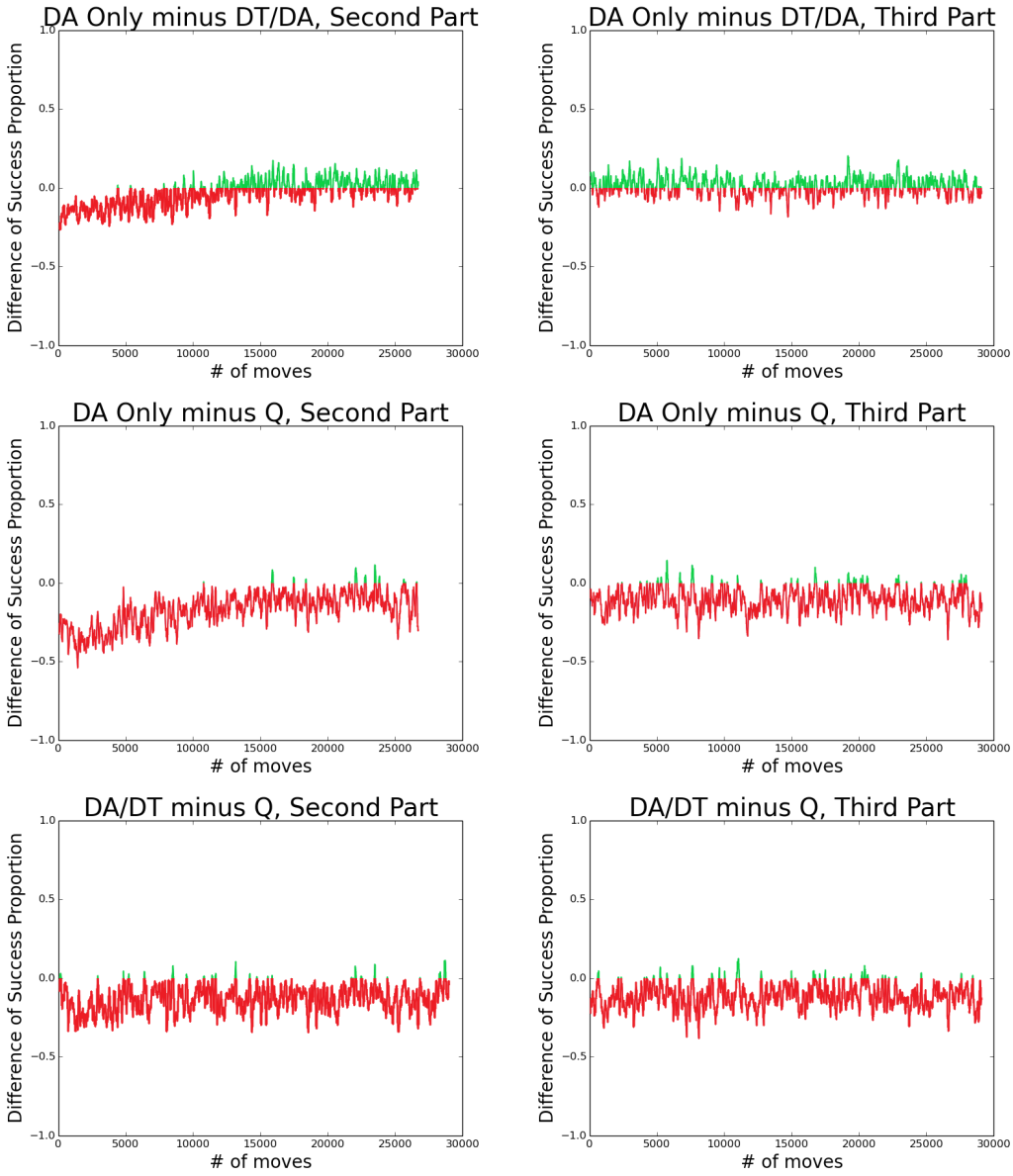

Figure 4.15: This figure shows the differences between the performance of different learner set ups in the second and third stages in Figure 4.14. The first column shows the differences of the second stage and the second column shows the differences of the third stage. The first row shows the difference between our learner using only **danger avoidance** mode and our learner using both **danger training** mode and **danger avoidance** mode. The second row shows the difference between our learner using only **danger avoidance** mode and standard Q Learning. The third row shows the difference between our learner using both **danger training** mode and **danger avoidance** mode and standard Q Learning.

The results for the experimental run with the $P_D(s)$ values on the safe path set to random values between 0.03 and 0.15 are shown in Figure 4.14. Additionally the differences between the performance of each experimental set up are shown in Figure 4.15.

These results show that our learner does not outperform standard Q-Learning in this environment. This is due to a combination of two factors. The first is that even when the safe path is followed successfully it still results in a significantly lower success rate than in the prior example, this means that our learner has significantly less room for error.

The second factor is that, though every square along the safe path is safer than every square off of the safe path, the relative difference between the safest squares off of the path and the most dangerous squares on the path can be relatively minor. This means that our learner has a significantly higher chance of choosing to move off of the path even in **Danger Avoidance** mode. Note for instance in Figure 4.11 the difference between E5 and D6 is less than average so when the system chooses which direction to move while on the E6 square it is more likely than normal to diverge from the path.

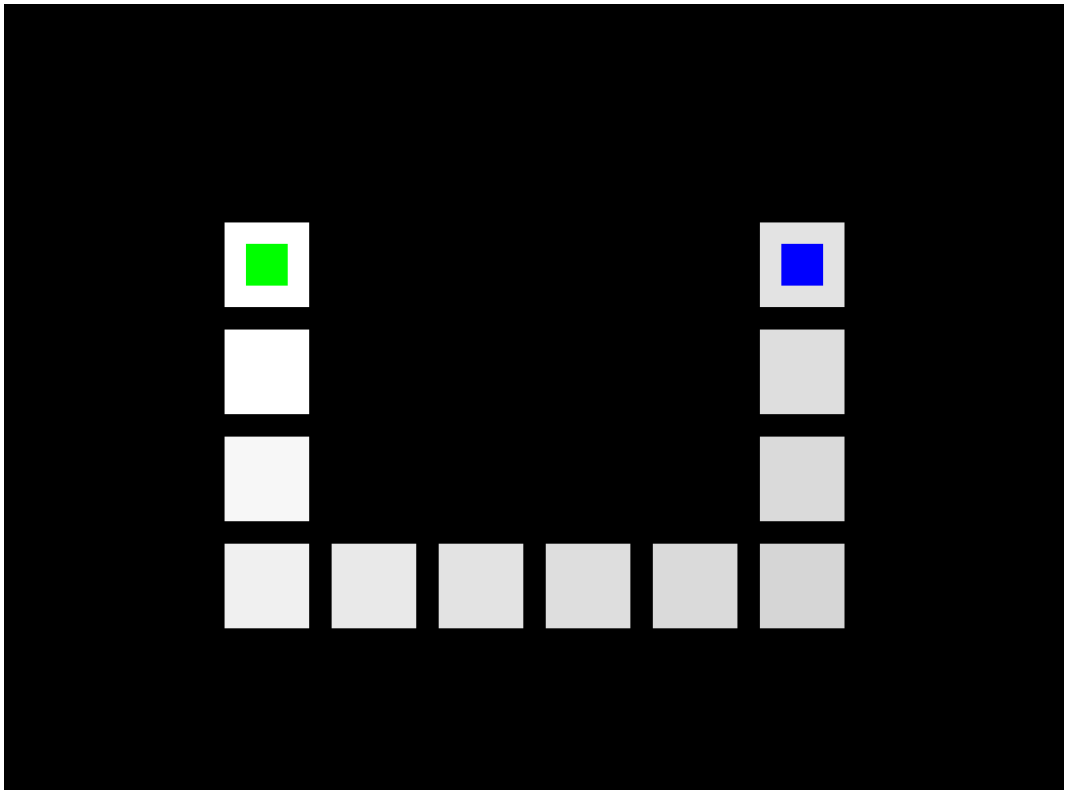

Figure 4.16: A heatmap of the Danger Avoidance mode values of each square on the Simple safe path example world after 2 million training steps.

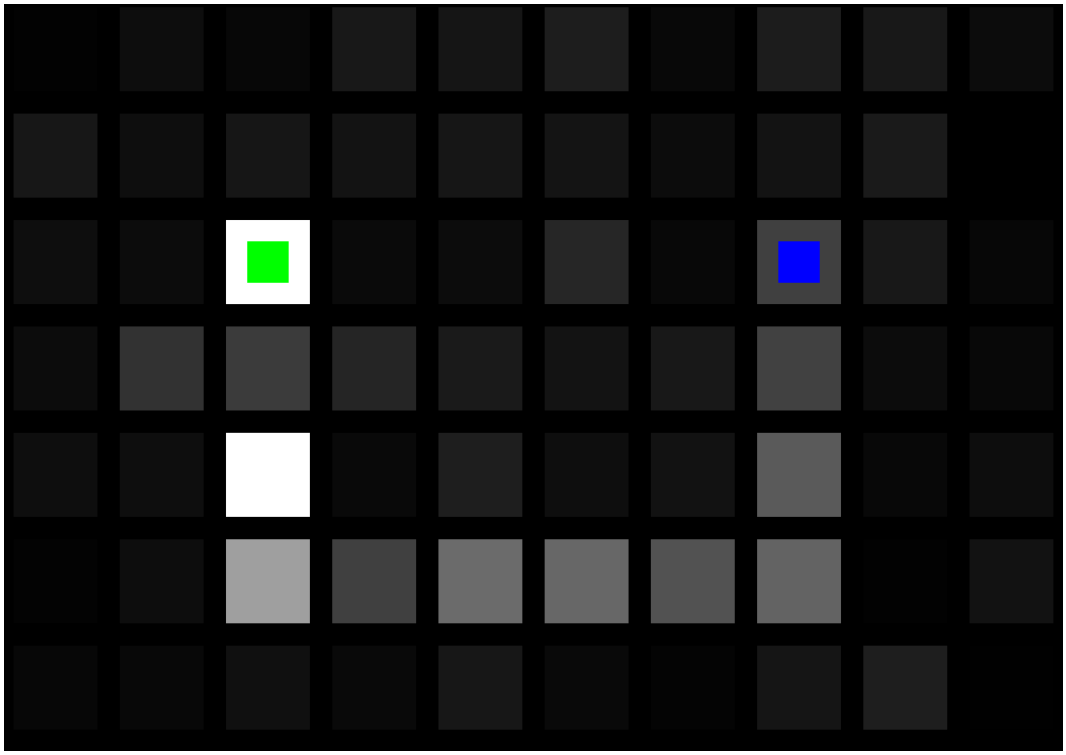

Figure 4.17: A heatmap of the Danger Avoidance mode values of each square on the random safe path example world after 2 million training steps.

This is further illustrated in Figures 4.16 and 4.17. These show the Danger Avoidance values for each state in the Simple Safe Path and Random Safe Path worlds respectively after 2 Million training steps, with brighter states were visited more often. The path is much more likely to be correctly followed in the Simple Safe Path example.

Next we will examine the results of a random safe path with a lower level of danger along the safe path.

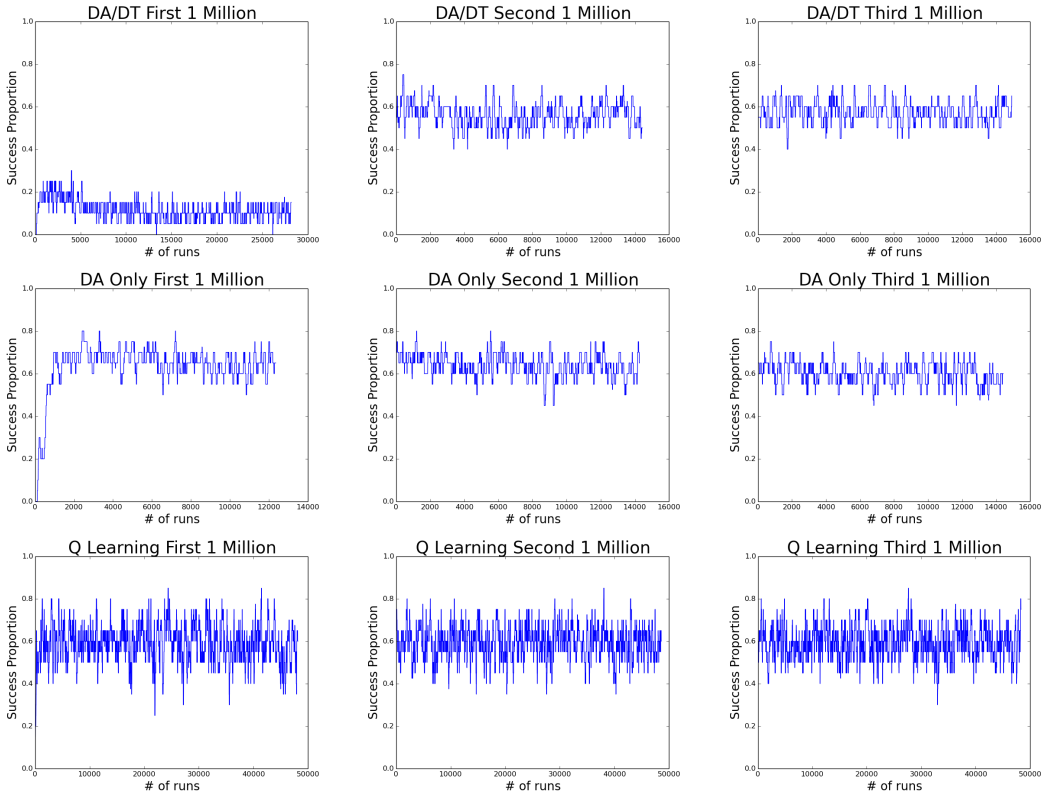

Figure 4.18: The proportion of runs that succeed with danger values on the safe path set to a low random value in a 20 run moving window which then has a median filter applied to it. Each graph represents a million steps with each experimental run composed of three million step stages, two in training and the third in exploitation. The first row is our learner using both **danger training** and **danger avoidance** mode, switching between stage 1 and 2. The second row is our system using **danger avoidance** mode only. The third row is standard Q-Learning. Note that though each graph represents the same number of steps the number of runs will differ from graph to graph as not all runs are composed of the same number of steps.

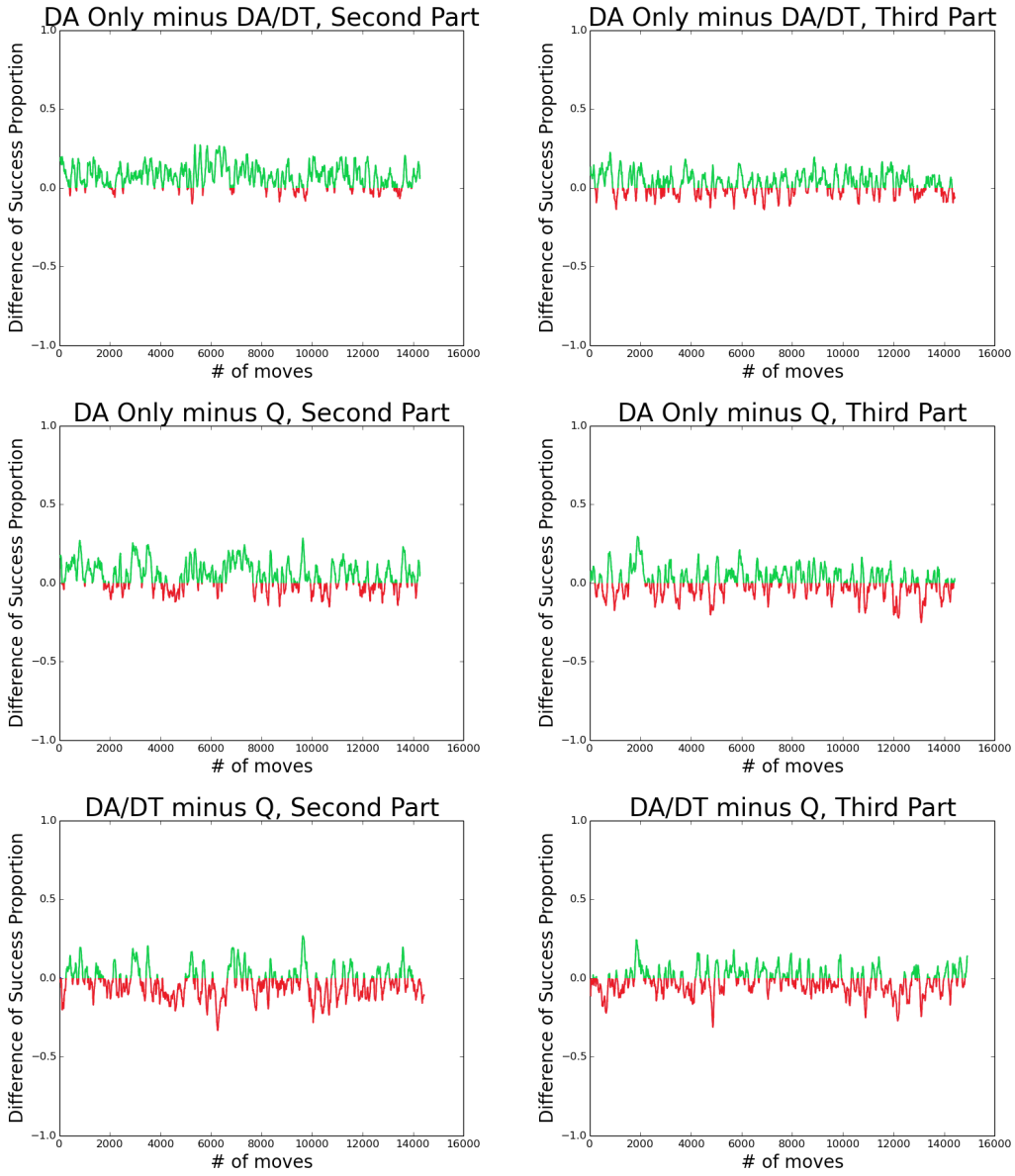

Figure 4.19: This figure shows the differences between the performance of different learner set ups in the second and third stages in Figure 4.18. The first column shows the differences of the second stage and the second column shows the differences of the third stage. The first row shows the difference between our learner using only **danger avoidance** mode and our learner using both **danger training** mode and **danger avoidance** mode. The second row shows the difference between our learner using only **danger avoidance** mode and standard Q Learning. The third row shows the difference between our learner using both **danger training** mode and **danger avoidance** mode and standard Q Learning.

The results for the experimental run with the $P_D(s)$ values on the safe path set to random values between 0 and 0.03 are shown in Figure 4.18. Additionally the differences between the performance of each experimental set up are shown in Figure 4.19.

These results show that our system and standard Q Learning are roughly comparable in this example, with our system having a slight advantage. Our system suffers from the same problems as in the prior example but the lower level of danger along the path allows it to still perform adequately.

These experiments show that there are situations in which our system outperforms standard Q-learning even when random variations are introduced.

4.4 Tailored Example Experiments

4.4.1 Simple Safe Path Experiment

Setup

As shown in Figure 4.1 in this scenario there is a u-shaped region of safe spaces connecting the origin state to the goal state - all other states are dangerous. This represents a single safe path to the goal.

States that make up the dangerous region are maximally dangerous and states that do not make up the dangerous region are minimally dangerous.

This means that the chance of succeeding when cutting straight across between the start and the goal is 17.8% whereas following the path has a 73.7% chance of success, not taking into account the $(1 - P_M(s, a))$ chance for the system to enter the wrong state when attempting to make a move. There is thus a distinct advantage in moving via the path rather than attempting to cut across. This is a greater theoretical benefit than the one obtained in Section A.1.1. Unlike in that example where there were many safe paths to the goal here there is only one safe path. Even a single incorrect move by the agent, either by an incorrect choice by the learner or random incorrect movement chance, can expose the agent to significantly increased risk though the agent should be capable of recovering from the later provided it does not enter the failure state immediately. Even if the agent makes the correct choice each time there will still be some diffusion due to the stochastic actions.

Aim

In this scenario the system should learn to follow the path of the Safe Path, rather than moving straight across, moving over risky squares, like the greedy

approach.

Data Generation

We allowed our system to navigate this world for 10000 runs in **danger training** mode. As seen in Figure 4.20 we then generated a heatmap based on 1000 runs of the system in **danger avoidance** mode. Both sets of runs were performed in heuristic mode.

Result

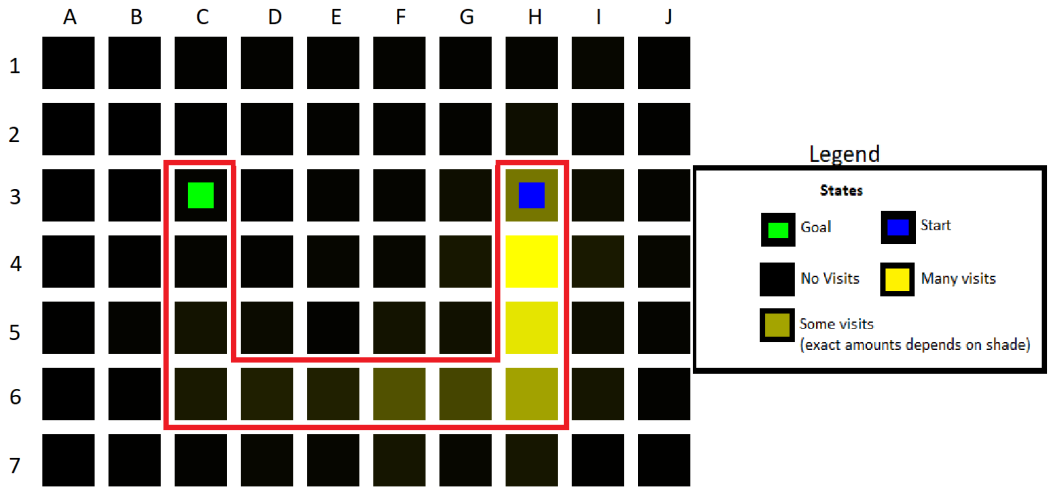

Figure 4.20: A heatmap the number of times each square was visited in 1000 runs of the simple safe path example in runtime mode after our system had performed with 10000 runs in the world in training mode. The system has a very strong preference for the safe path.

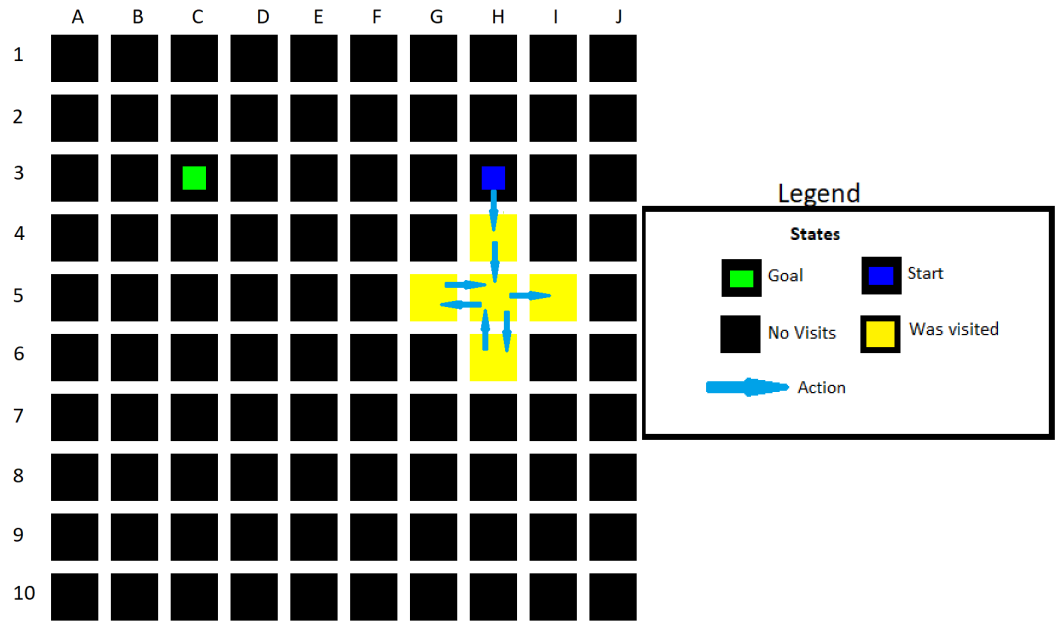

Figure 4.21: The runpath of a single run on the simple safe path world in runtime mode. Due to the probabilistic nature of movement in this environment the agent transitions into a dangerous square and transitions to a failure state.

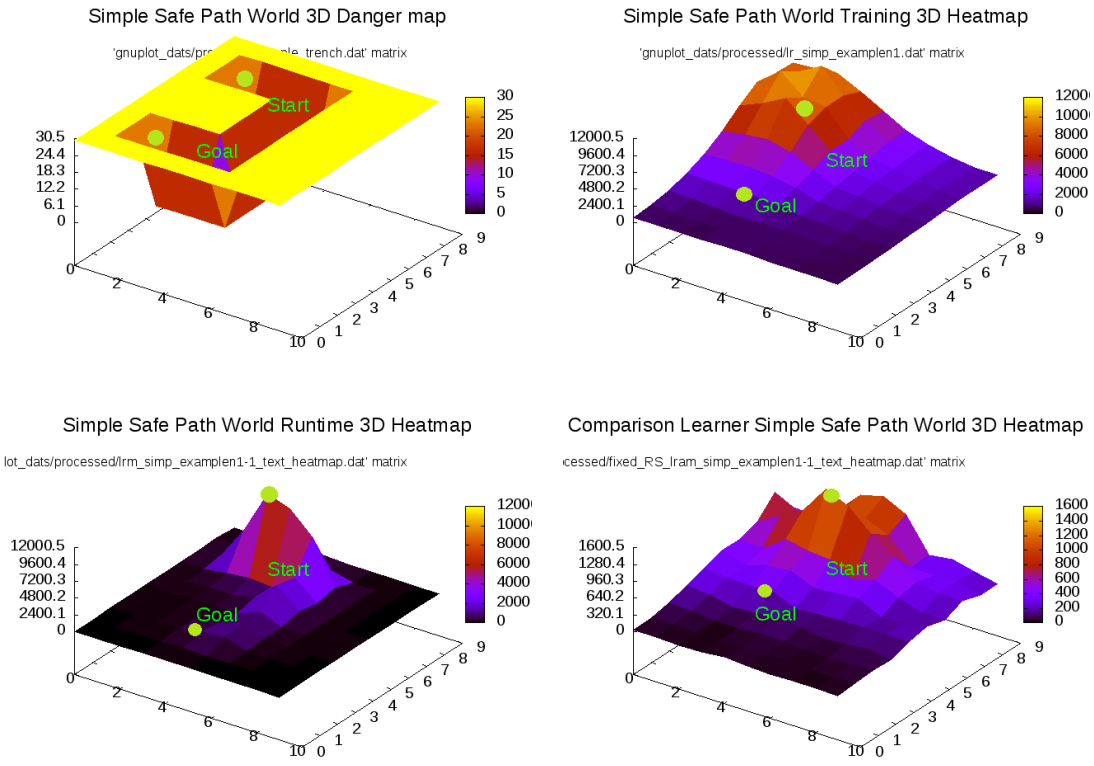

Figure 4.22: A 3D danger map of the Simple Safe Path World (Upper Left), a 3D heatmap of the 10000 training runs (Upper Right) and a 3D heatmap of the 1000 runtime runs in both our system (Lower Left) and the comparison learner (Lower Left).

Heatmaps of our system in both **danger training** and **danger avoidance** mode and the comparison learner are shown in Figure 4.22.

Each action is taken probabilistically with greater weight being given to better actions. In **danger avoidance** mode our system has a clear preference for following the safe path whereas the comparison learner does not.

The path chosen by the agent in any given run however is not perfect due to a couple of factors. As previously mentioned, at each stage the system determines its action probabilistically so even if the system has a clear preference for following the path there is a chance the system will, by probabilistic selection, choose to deviate from the path. This results in both the dangerous states around the start state being visited and the reduction in visit to states on the path as it progresses further. The first as, due to the number of times these states are visited, even unlikely choices will be taken with enough actions and the latter because as the system proceeds further down the path the probability of the system randomly

choosing to step off the path increases.

Also shown in Figure 4.21 is the runpath of one run by our learner in **danger avoidance** mode. The system shows a preference for the path but due to the probabilistic nature of the moment in which an agent has a $(1 - P_M(s, a))$ chance of moving to an incorrect state the agent still ventures off of it. Additionally while the agent has a strong preference for staying on the path it does not have a strong preference for moving in the correct direction along the path, causing it to stall. There, due to the high danger level associated with squares off the path in this example, it transitions to a failure state.

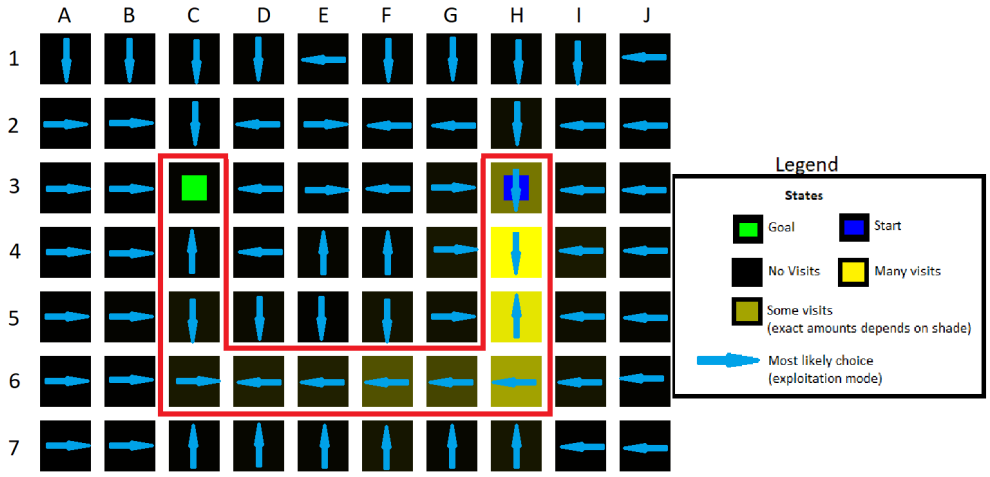

Figure 4.23: The most likely action by our system in runtime mode for each square on the simple safe path gridworld backed by a heatmap of 1000 runtime mode runs.

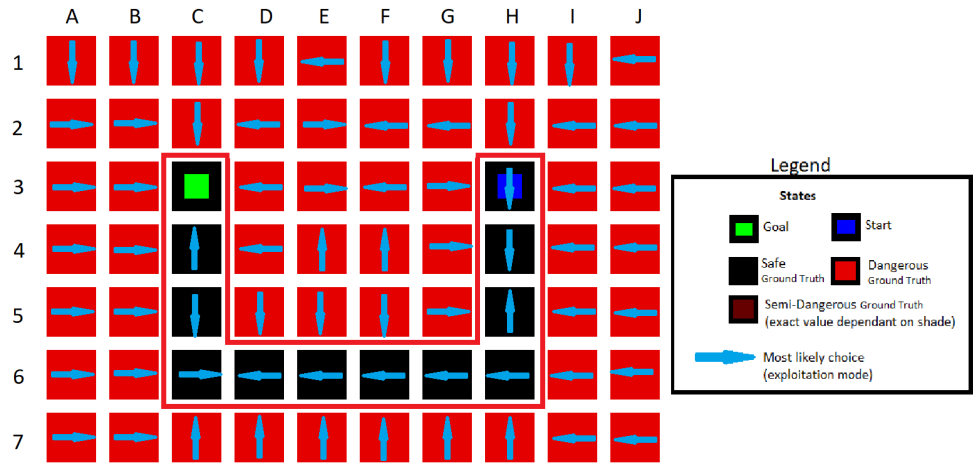

Figure 4.24: The most likely action by our system in runtime mode for each square on the simple safe path gridworld backed the danger value of each square.

The most likely choice in each square is shown in Figure 4.23 and 4.24. The system shows a clear preference for staying on the safe path in all path or path adjacent squares but may take unnecessary actions due to the safe path containing loops of likely actions. These loops are caused by the most likely actions in $C5$, $C6$ and $H5$ pointing in the incorrect direction along the path. This is likely caused the system prioritising reducing danger over moving the correct direction along the path. Although each of these states is relatively safe there is still a 3% chance of failure at in each square and if, by chance, one of these squares accumulates more failures than the last square on the path then the system may prioritise moving to the also safe last square on the path over moving forward.

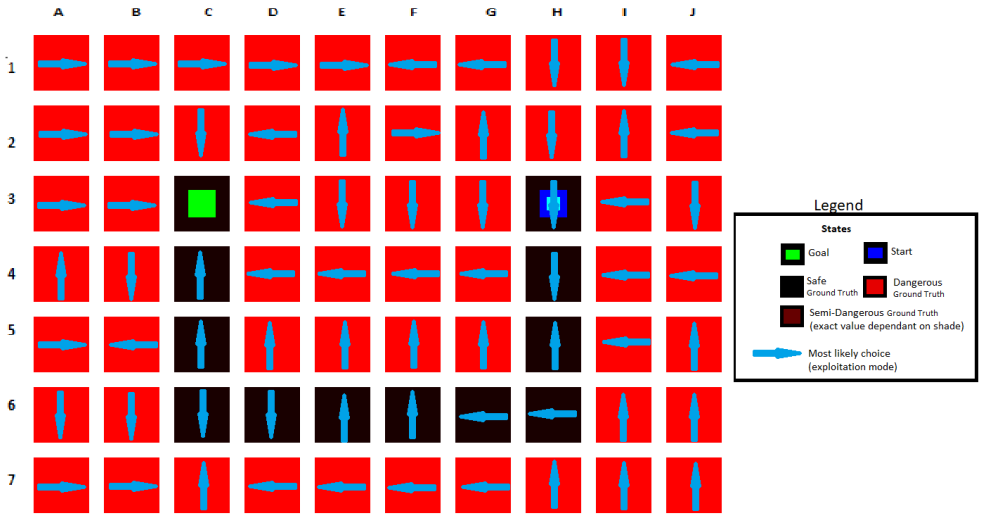

Figure 4.25: The most likely action by the comparison learner during runtime for each square on the simple safe path gridworld backed the danger value of each square.

For contrast the most likely choice in each square for one run of the comparison learner is shown in Figure 4.25. The comparison learner (the system described by Shen et al. (2014)) shows significantly less preference for the path than our system. As it only registers failure as a negative reward 35% of the time which is then averaged with the positive rewards when the system does not experience a failure 65% of the time the system does not see as clear a distinction between the direct but dangerous route and the safe but indirect route.

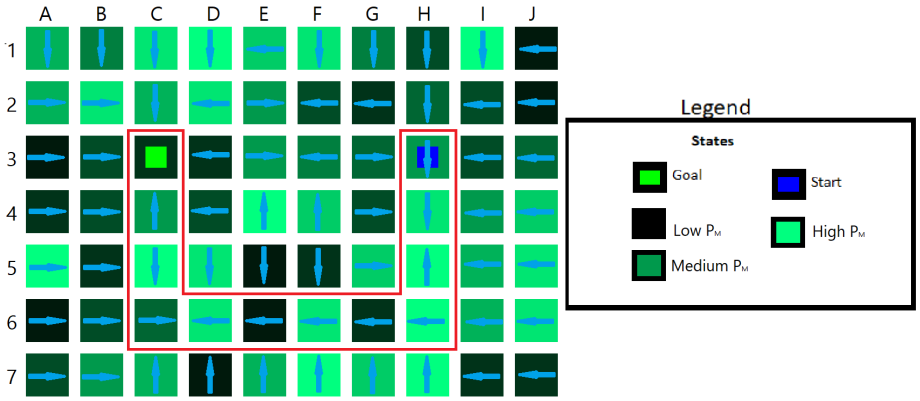

Figure 4.26: The P_M values of each square within the simple safe path gridworld with the most likely action overlayed.

4.4.2 Random Safe Path Experiment

Setup

This example, as shown in Figure 4.11, this example has the same safe path structure as Figure 4.1.

Instead of all safe squares being equally safe and all danger squares being equally dangerous safe squares will have a 3% and 15% probability of causing a failure state when entered while dangerous states have a 18% to 35% probability of causing a failure state when entered. This means that all dangerous squares will be more dangerous than all safe squares but the exact difference will vary.

Aim

The aim of this example is to show that our system still follows the path when random danger variation is introduced into the gridworld.

Data Generation

We allowed our system to navigate this world for 10000 runs in **danger training** mode. We then generated a heatmap based on a further 1000 runs of our system in **danger avoidance** mode. Both sets of runs were performed in heuristic mode.

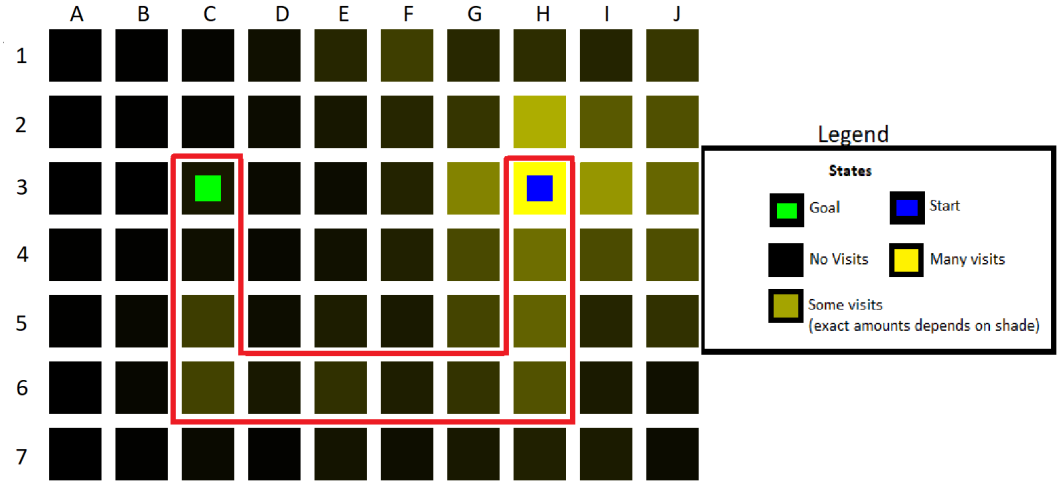

Figure 4.27: A heatmap of the number of times each square was visited in 1000 runs of the random safe path example in runtime mode after our system had performed with 10000 runs in the world in training mode. Our system still follows the path despite the added random danger variation.

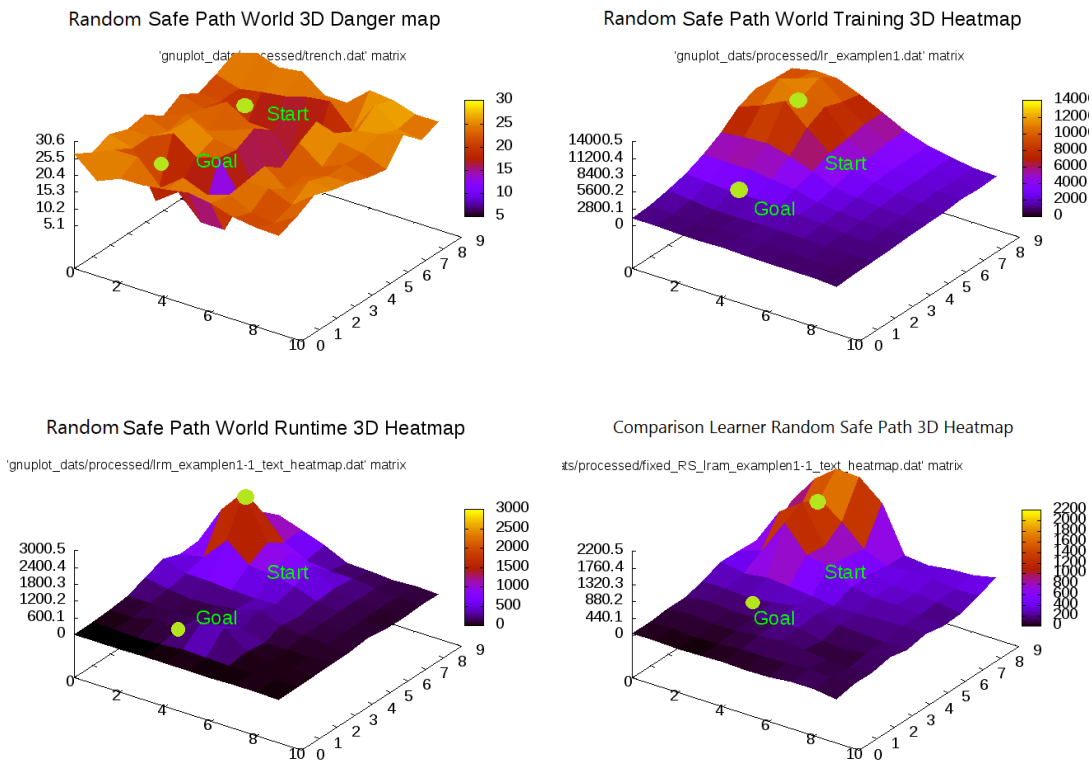

Figure 4.28: A 3D danger map of the Random Safe Path World (Upper Left), a 3D heatmap of the 10000 training runs (Upper Right) and a 3D heatmap of the 1000 runtime runs in both our system (Lower Left) and the comparison learner (Lower Right).

The heatmaps for this gridworld are shown in Figures 4.27 and 4.28 The path in **danger avoidance** mode weakens more quickly in this example because, while all the factors present in the previous example are still present, there is an additional factor. Unlike in the previous example though, squares in the path necessarily have lower risk associated with them than squares off the path, they do not necessarily have minimum risk. As such some runs will end with our system transitioning to a failure state while still on the path, reducing the number of runs in which our system gets to the end.

However despite these factors the path is still clearly favoured over the surrounding space.

The fact that C6 is favoured over F6 despite being later in the path shows the impact of randomness in the example. Agents that have left the path will be more likely to rejoin it at this point due to this state's relative safety. Additionally as it is safe compared to its surrounding the agent will be more likely to stall

temporarily in this area.

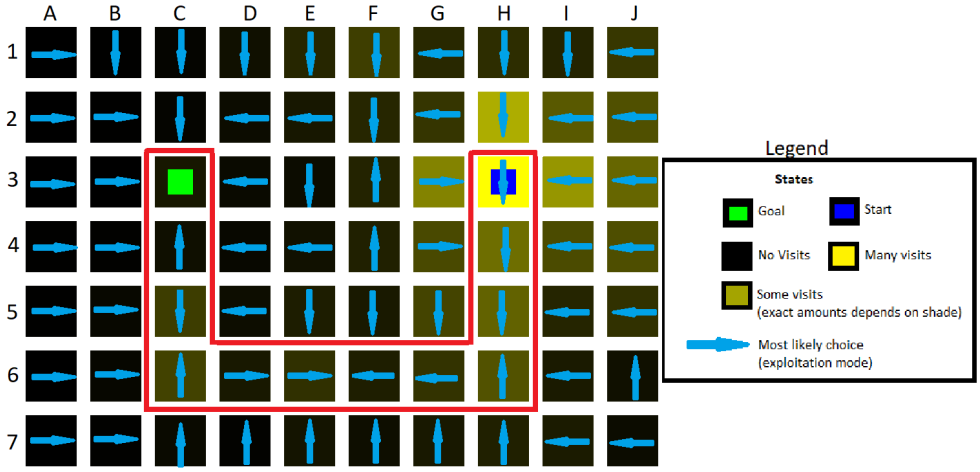

Figure 4.29: The most likely action by our system in runtime mode for each square on the safe path gridworld backed by a heatmap of 1000 runtime mode runs.

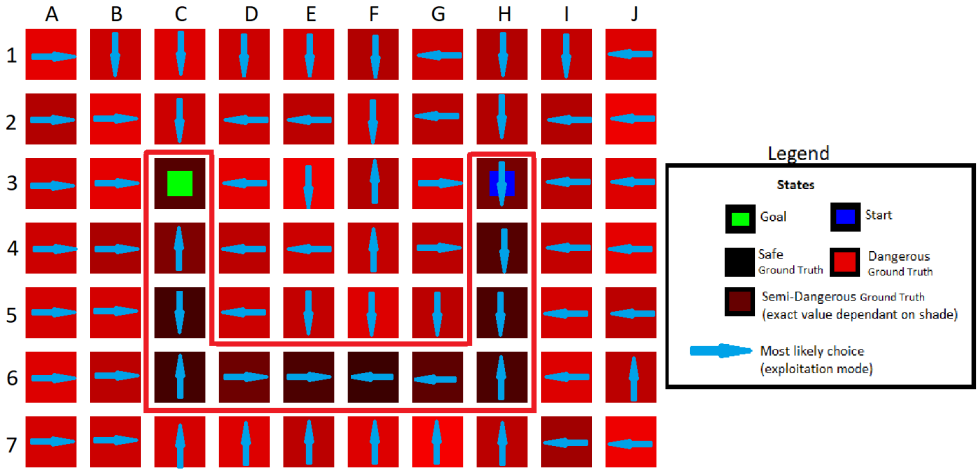

Figure 4.30: The most likely action by our system in runtime mode for each square on the safe path gridworld backed the danger value of each square.

The most likely choice in each square is shown in Figure 4.29 and 4.30. As with the previous example it shows a clear preference for the safe path with unnecessary actions due to loops present in the path.

4.5 Basic Quantitative Comparison Experiment

Set Up

A simple comparison of the relative successes of both systems on the Safe Path worlds (as well as the Dangerous Region worlds which will be explained in Section A.2) is shown in Figure 4.31 and Table 4.1 - our system represented by the red bars and the comparison learner represented by the green bars. These examples are made specifically to show the benefits of our system and so represent greater differences than most real world conditions.

Aim

Determine our learner's success rate compared to both the comparison learner and a greedy approach in heuristic mode.

Data Generation

First each learner was allowed to explore the gridworld for 10000 runs - our system doing so in **danger training** mode and the comparison learner doing so with parameters set to benefit exploration. Each learner was then made to attempt another 1000 runs of the gridworld - our system doing so in **danger avoidance** mode and the comparison learner doing so with parameters set to benefit exploitation.

Results

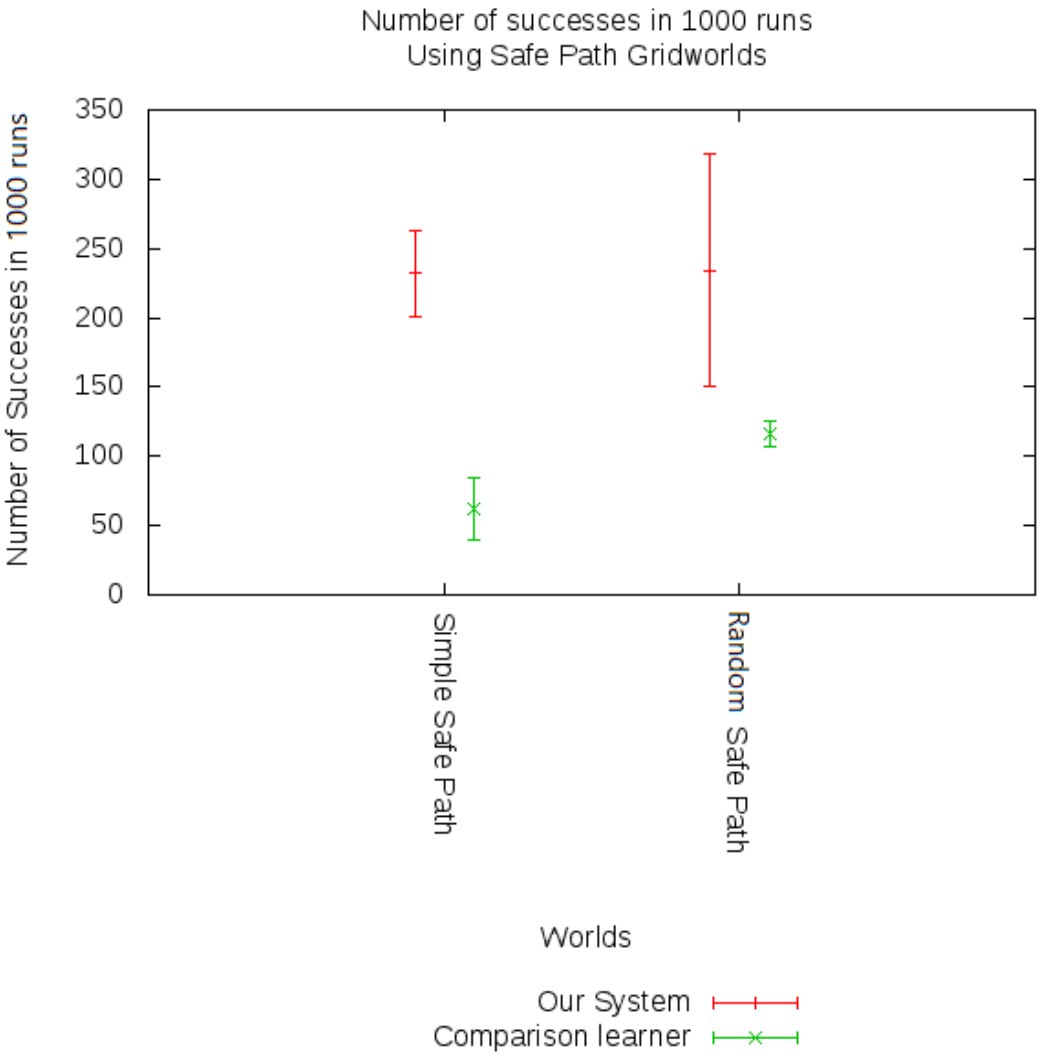

Figure 4.31: A basic comparison of the results of the two learners on these example gridworlds. Error bars represent standard deviation - each learner's results offset only for readability. In these examples our systems perform significantly better than the comparison learner but in the safe path examples show significant variance.

World	Our System	Comparison Learner	Straight Line
Random Safe Path	256	83	686
Simple Safe Path	179	39	598

Table 4.1: A comparison of the performance of both systems on the safe path worlds.

Every time the gridworld moves to a new square it potentially triggers transition to a failure state, the likelihood determined by how dangerous the state is.

In randomly generated examples the goal will not appear with less than two squares separating it from the start location.

The graph shows the number of successful runs each system made in the 1000 tries. While it is clear that the safe path example is more prone to failure than the danger region example as expected our system performs more effectively than the comparison learner. In both of these examples there is a clear safe path or paths and there is a clear dangerous region that an agent will move through if it takes the direct path. As such our system is able to navigate around this obstacle or follow the safe path whereas the comparison system is less likely to do so.

Overall these results show that our system can produce significant advantages over the comparison learner in many scenarios, particularly where there exists a safe path between the start and the goal that allows our system to exploit its advantages, but it is not inherently superior in all scenarios. In Section A.2 we will explore applying this same experimental set up to other worlds including the Dangerous Region worlds.

4.6 Summary

We performed a number of experiments examining the impact of our separate risk and reward metrics and the fashion in which they are utilised to create a policy.

We showed how our learner compared to standard Q Learning in the Simple Safe Path world, showing that our learner was able to detect and follow an indirect safe path to the goal that standard Q-Learning was not following, giving our system a higher success rate.

Next we altered the amount of training runs available to both learners in order to examine the impact on performance and when to switch from **Danger Training** mode to **Danger Avoidance** mode during training. While Q Learning is more effective given sufficiently limited training examples our system outperforms it after 150K training steps. Additionally the system ceases to improve after 250K **Danger Training** mode training steps, suggesting this as an ideal transition point.

We showed how our learner models the dangers present within the environ-

ment in both of the Safe Path worlds and showed how this can impact performance.

Finally we examined the effect of the relative weighting between reaching the goal and avoiding dangerous squares. This showed that the ideal weighting is dependent upon the application as there was not a consistent ideal weight shared between the various worlds tested.

We will now examine the danger exploration performance of the system with limited training data in detail.

Chapter 5

Risk Sensitive Exploitation with Uncertain Risks

The second novel contribution is how our system is able to handle not only uncertain outcomes due to risks but also uncertain risks due to limited training data.

In some cases the number of training runs than can be performed will be limited by factors such as high computational costs for training runs or even the ability to perform a limited number of training runs with cheap, disposable robots in a real environment. Our system is capable of using these limited runs to more efficiently learn the risk-reward landscape of a world, giving an advantage over other systems in certain use cases.

In a system such as this gridworld in which even the riskiest action is still only associated with a less than 40% chance of failure an action's true level of risk may not be apparent from a limited number of exploratory actions.

Our system is able to mitigate this effect as, unlike traditional risk sensitive reinforcement learning systems, it keeps track not only of estimates of how beneficial and risky a given action is ($Q(s, a)$ and $D(s, a)$ respectively) but also estimates of the level of certainty our system should have in these assessments ($C_Q(s, a)$) and $D(s, a)$) respectively.)

Using these certainty values we can explore more comprehensively during training but also avoid uncertain actions in **danger avoidance** mode even if they appear to be slightly safer. An action with an apparent 5% chance of failure that has been tested 50 times may be a more conservative choice than an action with an apparent 3% chance of failure that has been attempted only 12 times.

5.1 Quantitative Comparison with Limited Data Experiment

Set Up

In this experiment we will repeat the experiment performed in Section 4.5 but drastically limit the number of runs in both the training and exploitation phase.

Aims

This will allow us to compare the performance of our system and the comparison learner in a partially explored environments and furthermore the effect of different environment types in training runs.

Data Generation

In this experiment we will allow our system and the comparison learner (the system described by Shen et al. (2014)) to navigate each of the worlds used in the experiment described in Section 4.5 for only 50 runs in **danger training** mode before conducting a further 50 runs in **danger avoidance** mode and measuring the number of successes. The learner will continue to learn in *danger avoidance* mode, the model is not fixed. All runs will be performed in heuristic mode with a distance-to-goal heuristic.

Results

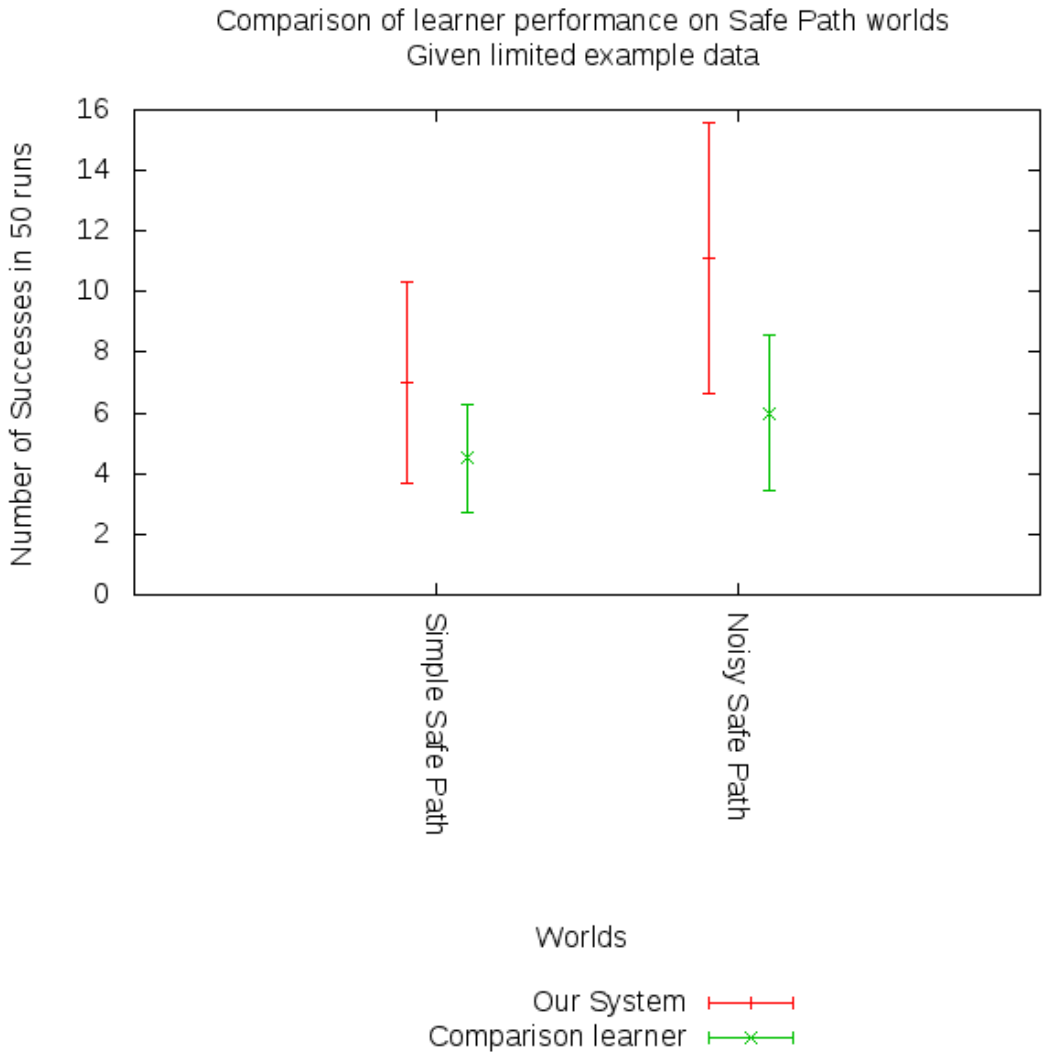

Figure 5.1: The average success rate per 50 runs for the safe path worlds given limited data. Error bars represent standard deviation - each learner's results offset only for readability. Our system still outperforms the comparison learner but to a lesser extent than in the high training data experiments and with greater variance.

World	Our System	Comparison Learner
Simple Safe Path	7	4.5
Random Safe Path	11.1	6

Table 5.1: The average success rate per 50 runs for the safe path worlds given limited training data.

As shown in Figure 5.1 and Table 5.1 our system has a slight advantage over the comparison learner in all these scenarios when the training time is limited, showing that our system is able to explore more effectively and make better decisions with limited data.

The examples have larger standard deviations than in Section 4.5. This is likely because the larger number of training runs allows these examples to reach a reach a consistent ceiling on model effectiveness whereas with the lower number of training runs the random choices in each individual training run can lead to differences in performance.

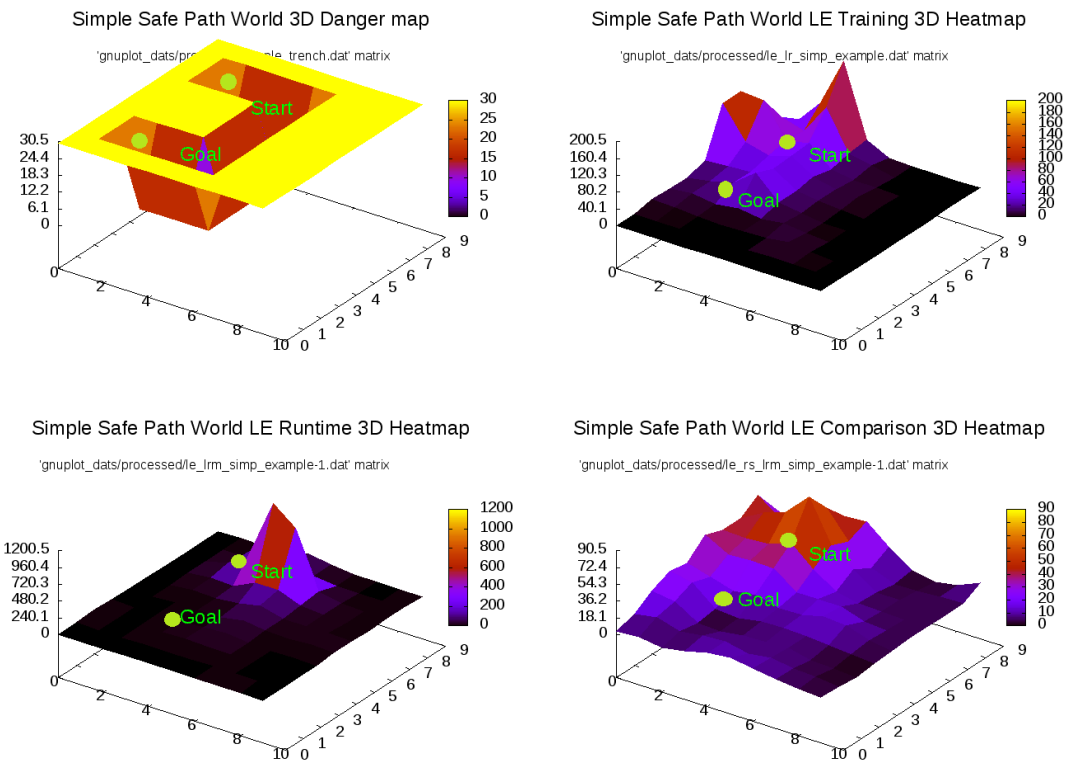

Figure 5.2: A 3D danger map of the Simple Safe Path World (Upper Left), a 3D heatmap of the 50 training runs (Upper Right) and a 3D heatmap of the 50 runtime runs in both our system (Lower Left) and the comparison learner (Lower Right).

The heatmaps of limited training and exploitation based on limited training data by our system and the comparison learner in the Simple Safe Path World are shown in Figure 5.2. Note that the structure of the safe path is clearly visible within our system's exploitation heatmap, after the training heatmap traced the boundaries of the dangerous region the agent needed to avoid. The comparison learner on the other hand attempts to move directly to the goal.

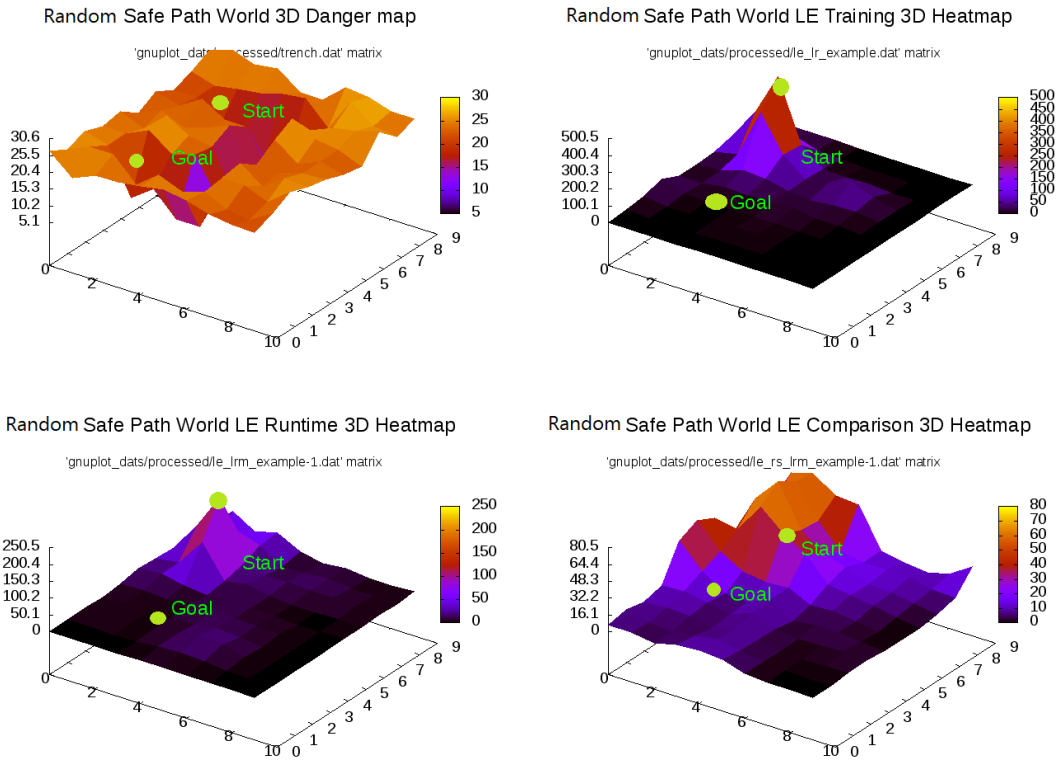

Figure 5.3: A 3D danger map of the Safe Path World (Upper Left), a 3D heatmap of the 50 training runs (Upper Right) and a 3D heatmap of the 50 runtime runs in both our system (Lower Left) and the comparison learner (Lower Right).

The heatmaps of limited training and exploitation based on limited training data by our system and the comparison learner in Random Safe Path World are shown in Figure 5.3. Although still not successful the majority of the time our system has again identified the safe path structure of the world, which is visible within the exploitation heatmap, even given the limited training data. The comparison learner in contrast does not take the safe path structure into account and attempts to reach the goal more directly by crossing the dangerous region.

5.2 Effect of Relative Weights on Exploration

Set Up

In this experiment we alter the relative weights of $Q(s, a)$ and $D(s, a)$ during the training phase with the system in **danger avoidance** mode in the Simple Safe

Path world. Additionally the same experiment was performed on two randomly generated worlds as will be discussed in Section A.6.

Aims

This will allow us to determine to what extent it is better to explore with a focus on determining the best path compared to a focus on exploring potential risks.

Data Generation

In this experiment we will allow our system and the comparison learner to navigate each of the worlds used in the experiment described in Section A.3 for only 50 runs in **danger training** mode before conducting a further 50 runs in **danger avoidance** mode and measuring the number of successes. Our learner will operate in heuristic mode to reduce the effects of Q-Learning on the final results, this will result in the success rate being lower than if these runs were performed in secondary learner mode.

Results

Figure 5.4: The average success rate per 50 runs depending on the relative value between seeking the goal and avoiding risks in exploration mode given limited exploration runs. Error bars represent standard deviation - each learner's results offset only for readability. There does not appear to be a consistent improvement of altering the relative values in either direction.

Relative weights	Random 2	Simple Safe Path	Random 3
0.25	19.1	6.7	9.7
0.5	20.1	4.3	11.1
0.75	21.6	4.8	9.7
1	18.6	5.1	13.7
2	20.8	5.4	11.9
3	20.9	3.7	12.3
4	18	6.1	10.9
5	20.8	4.9	10.5
10	17.1	6	12.3

Table 5.2: The average success rate per 50 runs depending on the relative value between seeking the goal and avoiding risks in exploration mode given limited exploration runs.

The results of this experiment are shown in Figure 5.4 and Table 5.2.

In most of these examples the distributions are fairly even, showing decent performance with all weightings.

This suggests that even in examples where one of these quantities is heavily weighted over the other the incidental collection of data about the other value is enough to allow our system to perform adequately.

However in each of the worlds tested the maximum values are either at a relative weight of around one or an even distribution, with the exact value depending on the environment

This suggests shows that taking both $Q(s, a)$ and $D(s, a)$ into account in the exploration maximizes our system's chances of successfully producing a useful model of the environment when given limited exploratory runs.

5.3 Summary

We performed a number of experiments examining the ability of our system to operated when the quantity of training data is limited.

We performed an experiment to determine the impact of incrementally lowering the amount of training data our system has access to. It showed that our system is able to maintain a high degree of performance as the amount of training data is reduced as is shown in Figure A.36.

We performed an experiments comparing the quantitative performance of our system and the comparison learner under limited training data conditions in both the 8 worlds used in Section 5.1 and the fixed dangerous square percentage worlds used Section A.5. Our system performed better than the comparison learner in all scenarios and to a greater degree than the experiments with large amounts of training data attempted in Sections 4.5 and A.3.

We measured the effects of changing the relative weights of goal seeking and avoiding danger for both training and exploitation when our system is only given limited training data. As with altering the relative weights with plentiful training data the ideal weighting is dependent upon the application.

Chapter 6

Conclusion

In many applications an agent will need to complete a task in an environment while also avoiding dangers. For instance if a robot makes the wrong move while driving over a pile of rubble it may slip, fall and become damaged. We aimed to create a learner capable of responding to the dangers present in its environment, learning about them during training and then avoiding them during exploitation while still pursuing the goal.

We performed our experiments in a gridworld environment with static start and goal locations. Each time the agent transitions to a new state it has a $P_D(s)$ probability of entering a failure state where $P_D(s)$ is in the range 0 to 0.35. By associating the risk of failure with a particular state rather than a particular action we create an environment in which the danger associated any particular location is easily quantifiable. As such it is simple to determine how well each system models the dangers present in the environment. We also assume that the agent will not be penalized for failures during training but will be penalized for failures during exploitation.

We selected two comparison learners: standard Q-learning and a variant of Q-Learning described by Shen et al. (2014). The former is a well known and well characterized reinforcement learning system used in a variety of situations. The latter has been demonstrated to operate similarly to human risk evaluation behaviour and as such acts a proxy for an average, non-expert human evaluating the same evidence encountered by our system.

We have presented a novel system that deals with danger present within an environment. We have also tested our system in a range of scenarios in heuristic mode, in which our learner combines learning and then avoiding the dangers present with in the environment with a greedy approach to finding the goal. We

then performed experiments within an environment with our system in what we call secondary learner mode, where our system works better than conventional approaches. Our learner also learns how to avoid the dangers present in the environment but combines it with Q-Learning in order to find the goal. In this mode Q-Learning still runs as normal, learning after every action, but our learner adds a bias in action selection depending upon the danger and certainty values of each action.

We have demonstrated that our learner acts more effectively than standard Q-learning in some scenarios by navigating around direct but dangerous paths to the goal in favour of indirect but safe paths. In Section 4.1 we show that our system has a higher success rate than Q-Learning in the Simple Safe Path world in which a 'u' shaped path of safe squares connects the start location to the goal with all other squares being dangerous. This indicates that our system is able learn to stay on the safe path, which is the optimum path for balancing risk vs reward, faster and more often while standard Q-Learning, which is unable to distinguish between high-risk, high-reward options and lower-risk, more conservative options, cuts across straight to the goal given the same quantity of training steps. In situations where we want to limit risk, such behaviour is inappropriate and yet impossible to encode in a way that conventional Q-learning can make use of.

We have also showed the utility of **danger training** mode which attempts to maximise both danger and Q-value and minimise certainty when selecting actions. This allows our system to efficiently explore the dangers present in the environment provided that the agent is able to encounter failures during training without real-world consequences. It also allows the learner avoid a period of wasted training time that would have occurred using **danger avoidance** mode only. In **danger avoidance** mode the agent maximises the Q-Value and certainty and minimises danger. However when our learner runs in **danger avoidance** mode without having any prior information about the environment our learner remains in safe states near the goal rather than exploring the environment to attempt to locate the goal. This is because when the learner has no information about where the goal is located its only priority is remaining safe, causing it to repeat a small number of known safe actions repeatedly. Eventually the random aspect of action selection will make the agent encounter the goal and begin attempting to reach it but this may require many more steps.

We perform several experiments in which we train our learner in **danger training** mode to learn the dangers present in the environment and then train our learner in **danger avoidance** mode to learn the Q-values. This means that

the learner determines both which states are dangerous and which paths to the goal are beneficial during training. During exploitation our learner is able to use this information in **danger avoidance** mode to take actions that simultaneously avoid danger and allow the agent to reach the goal.

Standard Q-Learning does have an initial advantage over our learner when the amount of training is sufficiently limited, our learner must learn both Q-values and the $D(s, a)$ danger values requiring more exploration to understand the environment. However once our learner has converged it will be able to use its knowledge of the dangers present in the environment to avoid dangerous states with significantly fewer training runs than Q-Learning requires to converge on an optimal solution. In Section 4.2 we show Q-Learning outperforms our learner in the Simple Safe Path world when our learner has less than 100K training steps but our system consistently outperforms standard Q-Learning when it has more than 200K training steps. Based on these results we have selected 250K training steps as an empirically determined best transition point between **danger training** mode and **danger avoidance** mode for this environment.

Our learner is able to make the bulk of mistakes in the beginning during training when we've engineered the environment to make the mistakes inexpensive such as by performing the training runs in simulation. As long as a particular environment can be adapted to make failures inexpensive during training this learner can be applied to in order to make an agent that avoids danger while seeking the goal. One use case for which this applies is long standing disaster areas such as the Fukushima power station in which there are numerous dangers to the robot from rubble or debris but plenty of time to simulate the robot's actions using all available data. Another use case is that using this learner a robot could learn to manoeuvre over a steep mountainside in which the terrain presents an ever present risk of falling but the mountainside can be easily observed, scanned and simulated.

Future Work

Currently this system is limited to applications in which the agent is able to explore freely without consequences for transitioning to the failure state in training mode. This could be expanded to applications in which failures in training mode are cheap but non-free, for instance small, cheap 3D printed robots being sent into a hazardous environment ahead of a more expensive robot.

This could be done with a system that takes into account both eventual risk

and immediate risk and attempts to maximize eventual risk and minimize immediate risk in training - effectively take the actions that lead to failures but then do not taking the final step. This system would map the exact boundaries of dangerous areas while only falling victim to them occasionally.

Our system can also become stuck in regions of safe states surrounded by more dangerous states. Adding a mechanism by which actions that have already been taken in a given run are weighted negatively would help the agent break out of these regions faster and improve the performance of our system.

Another avenue of investigation is to alternate runs between **danger training** mode and **danger avoidance** mode during training. This would distribute training time evenly between exploring the danger present in the environment and generating accurate Q-values, allowing **danger training** mode to better prioritize determining the $D(s, a)$ values of states with a high Q-Value and **danger avoidance** mode to work out paths to the goal that avoid dangerous states simultaneously.

It may also be useful for the agent to compare the reduction in danger of taking a less direct action to the averaged danger in the agent's current region. If the region's general danger is high enough that taking more actions is more likely to cause a failure than taking a more direct, more dangerous path then the agent should move directly to the goal instead.

Investigating how the discounting of future rewards effects this approach may also lead to general improvements in performance.

Appendices

Appendix A

Other Experiments

A.1 Other Tailored Example Experiments

A.1.1 Simple Danger Region Experiment

Set up

As shown in Figure A.1 a region of eight dangerous squares stretching from *E3* to *F7* blocks the direct path between the start state from the goal. This dangerous region does not extend fully to the top or bottom of the gridworld - it can be avoided by going above or below.

States that make up the dangerous region are maximally dangerous and states that do not make up the dangerous region are minimally dangerous. Maximally dangerous states are defined as having a 35% probability of failure whenever the square is travelled to. Minimally dangerous states are defined as having a 3% probability of failure every time a square is travelled to.

This means that the chance of succeeding when cutting straight across between the start and the goal is 39.75% whereas the shortest path that does not cross a maximally dangerous square has a 73.7% chance of success, not taking into account the $\neg P_M(s, a)$ chance for the system to enter the wrong state when attempting to make a move. This effect will reduce the chance of success of both paths but the longer path will be affected more, reducing the comparative effectiveness of our system compared to the comparison learner.

Aim

In this example we aim to show that the system avoids the centre, dangerous region, going around the sides instead.

Data Generation

First the system is allowed to explore the gridworld for 10000 runs in **danger training** mode. It is then run an additional 1000 times in **danger avoidance** mode and a heatmap showing the frequency each square is visited is generated based on this **danger training** mode run - yellower squares being visited more often. Both sets of runs were performed in heuristic mode.

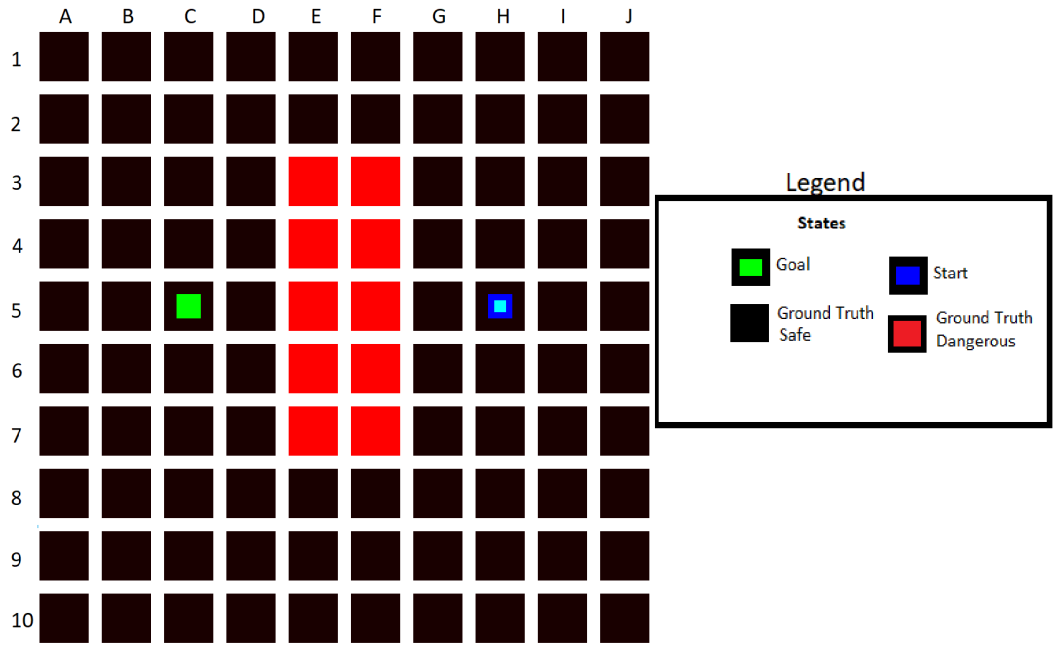

Figure A.1: The simple danger region example world. A small dangerous region separates the start location from the goal.

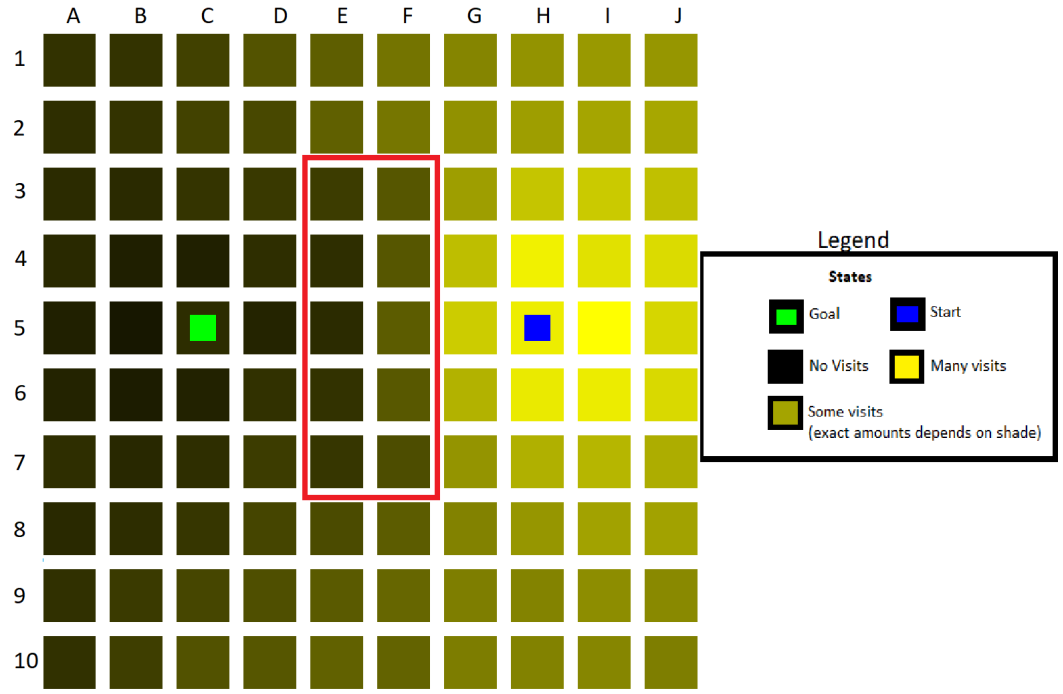

Figure A.2: A heatmap of the number of times each square was visited in 1000 runs using runtime mode with out system in the simple danger region example world after the system has explored in training mode for 10000 runs. The system avoids the dangerous region.

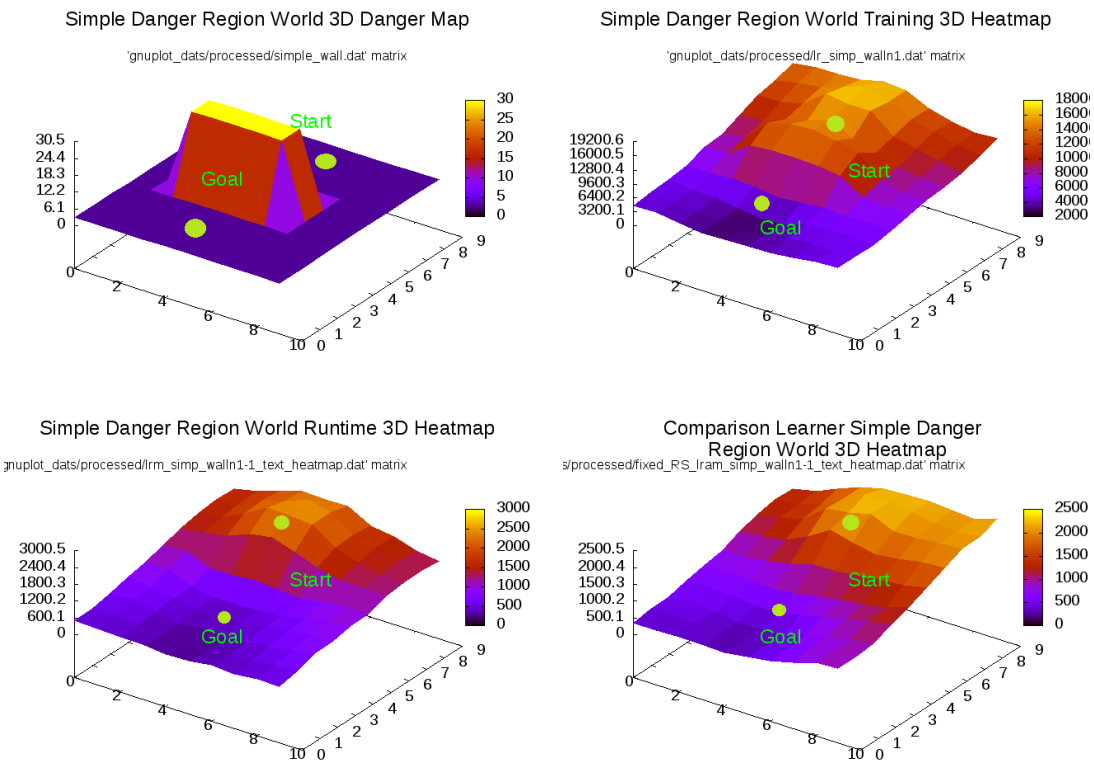

Figure A.3: A 3D danger map of the Simple Danger Region World (Upper Left), a 3D heatmap of the 10000 training runs (Upper Right) and a 3D heatmap of the 1000 runtime runs of both our system (Lower Left) and the comparison learner (Lower Right).

The resultant heatmaps are shown in Figures A.2 and A.3. Note the presence of the cavity in the centre of our system's **danger avoidance** mode heatmap and how the heatmap flows around it to the top and bottom. This shows the system exhibits the property of venturing around the large, dangerous region in order to become closer to the goal while avoiding danger. However given that the danger region is directly between the goal and the start square. This also means that the goal is also on the edge of this cavity region, causing less successes than would be ideal. This property is not shared by the comparison learner heatmap in which a significant density of moves overlaps with the dangerous region.

In **danger training** mode on the other hand our system specifically seeks and enters the dangerous region. This allows the agent to know the exact extent and size of the dangerous region.

As there is no specific reason to transverse any particular square as long as it's closer to the goal and not one of the dangerous squares the heatmaps

quickly diffuse as you move away from the start as there are no choke points to concentrate the agent in any particular location. As the actions are determined probabilistically even actions that are not beneficial such as moving away from the goal are chosen sometimes, albeit in lesser proportion.

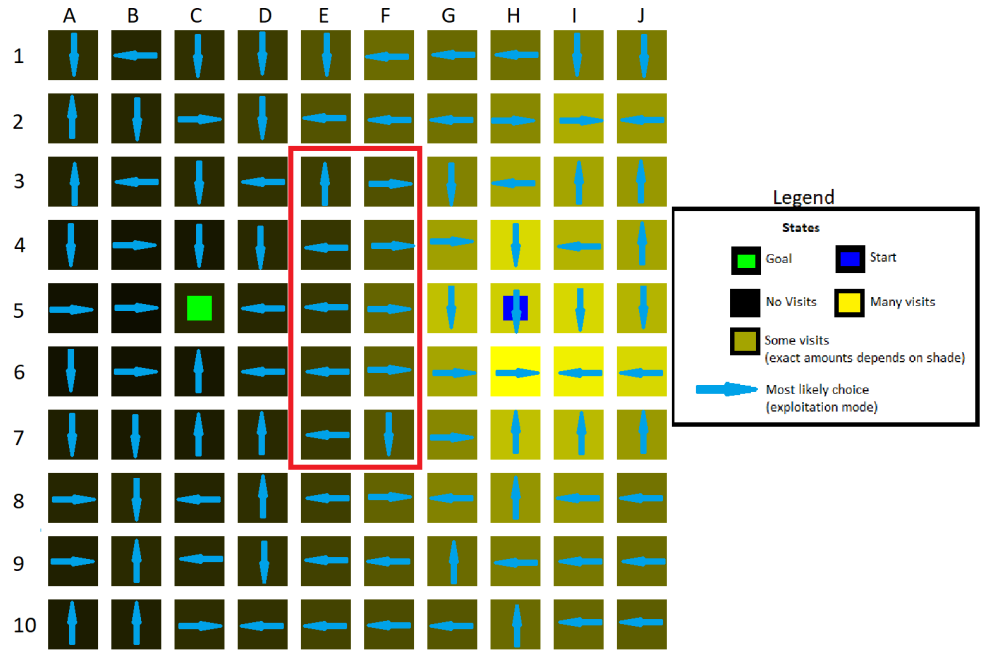

Figure A.4: The most likely action by our system in runtime mode for each square on the simple danger region gridworld backed by a heatmap of 1000 runtime mode runs.

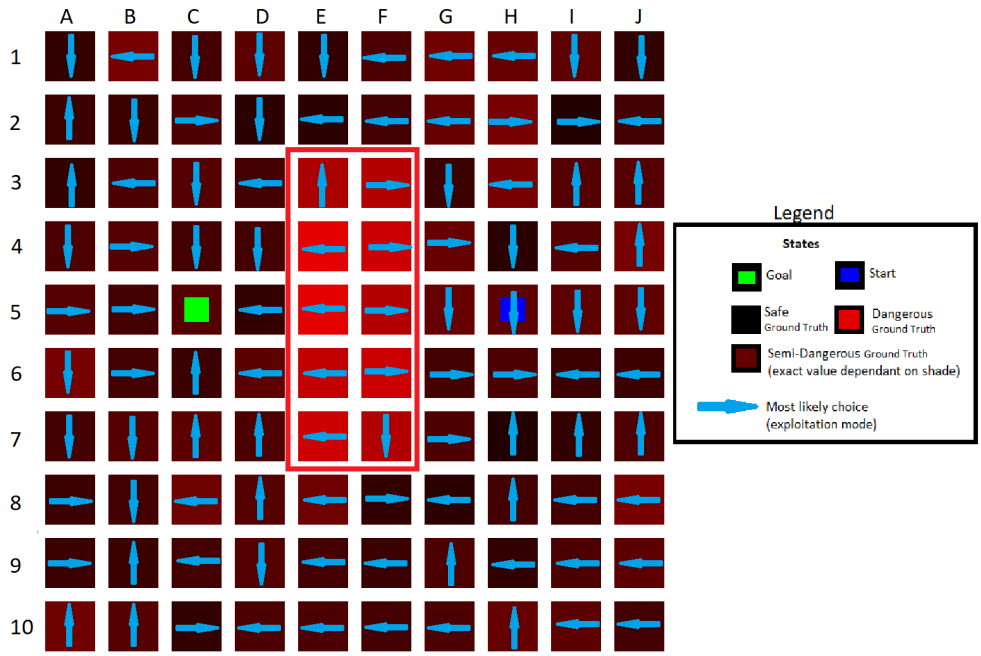

Figure A.5: The most likely action by our system in runtime mode for each square on the simple danger region gridworld backed the danger value of each square.

The most likely choice in each square is shown in Figure A.4 and A.5. The system effectively avoids the centre, dangerous region.

A.1.2 Random Danger Region Experiment

Setup

As shown in Figure A.6, like in the prior example in Figure A.1, a region of eight dangerous squares stretching from *E3* to *F7* blocks the direct path between the start state and from the goal. This dangerous region does not extend fully to the top or bottom of the gridworld - it can be avoided by going above or below.

Instead of all safe squares being equally safe and all danger squares being equally dangerous, as in the prior example, safe squares will have a 3% and 15% probability of causing a failure state when entered while dangerous states have a 18% to 35% probability of causing a failure state when entered.

Aim

The main of this example is to show that the prior example still holds when random variation in the danger values is introduced into the system. That the agent will select a path to the goal that does not enter the dangerous region.

Data Generation

As with the previous example we performed 10000 runs in **danger training** mode and then, as shown in A.7, generated a heatmap based on a further 1000 runs in **danger avoidance** mode. Both sets of runs were performed in heuristic mode.

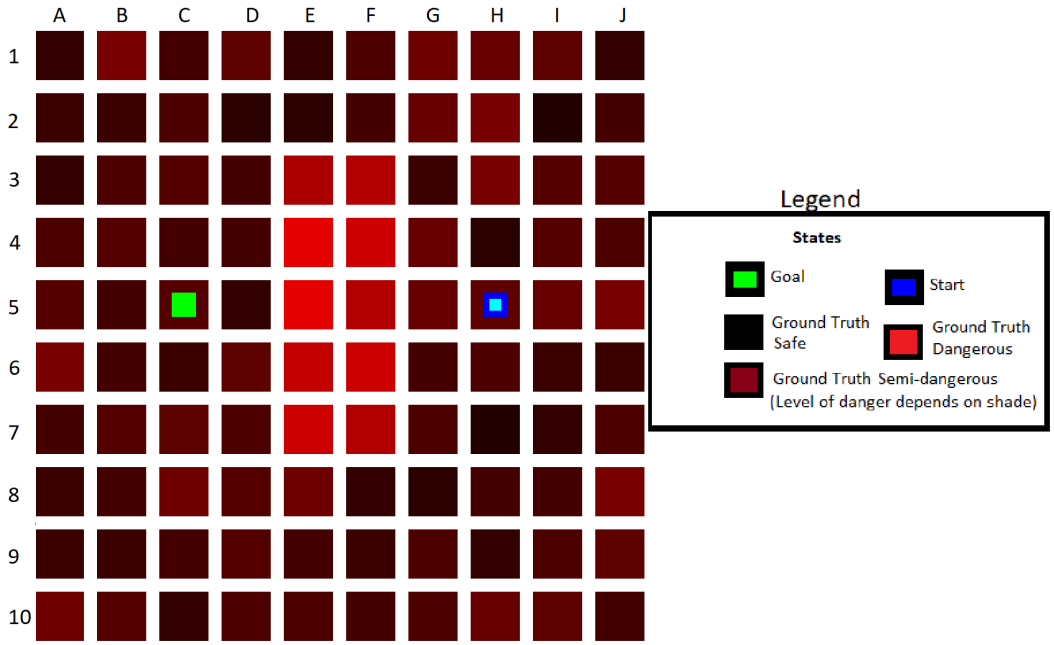

Figure A.6: The random danger region example world. The previous example but with added random variation in the danger values to make learning more difficult.

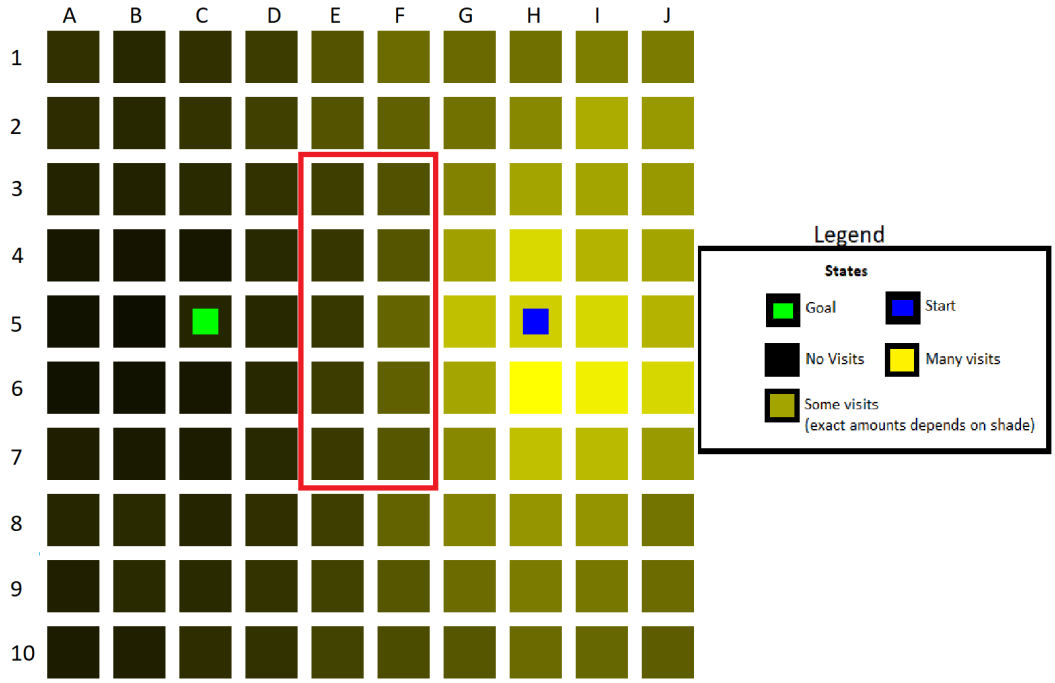

Figure A.7: A heatmap of 1000 runs using runtime mode with our system in the random danger region example world after the agent has explored for 10000 runs in training mode. Our system still avoids the dangerous region as in the last example despite the added random danger variation.

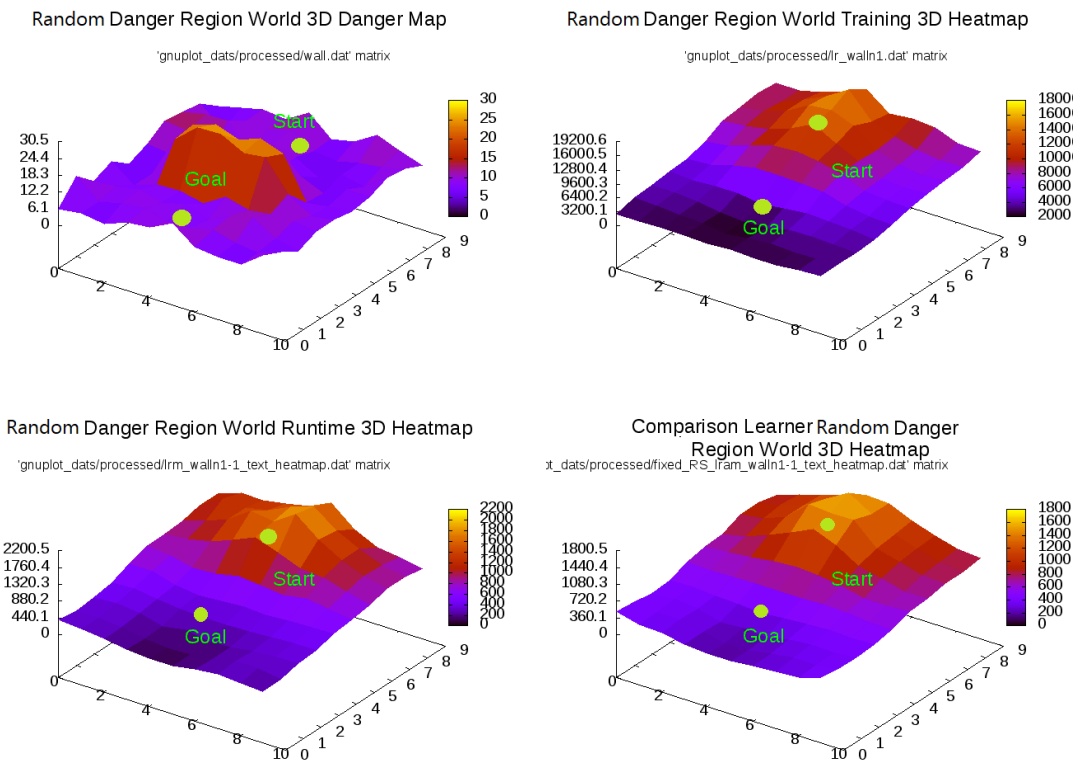

Figure A.8: A 3D danger map of the Random Danger Region World (Upper Left), a 3D heatmap of the 10000 training mode runs (Upper Right) and a 3D heatmap of the 1000 runtime mode runs in both our system (Lower Left) and the comparison leaner (Lower Right).

The same pattern of avoiding the centre to move around the top and bottom shown in A.2 is still present in this example. Similarly the same effect of diffusion as you move away from the goal and suboptimal actions being selected due to the probabilistic nature of the system is also present.

Similarly the comparison learner also behaves as it did in the prior example as shown in Figure A.8.

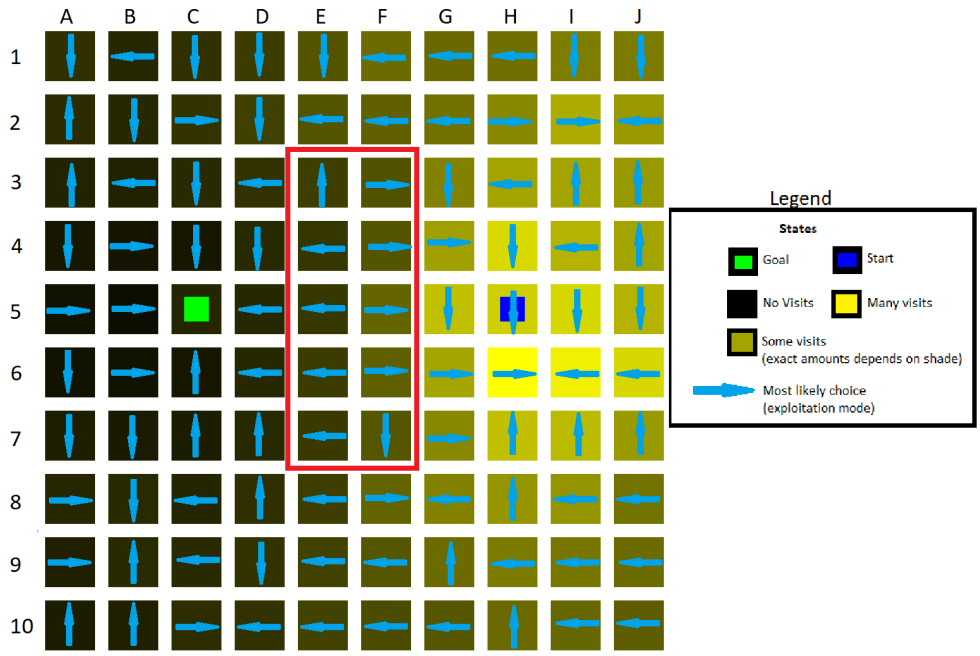

Figure A.9: The most likely action by our system in runtime mode for each square on the danger region gridworld backed by a heatmap of 1000 runtime mode runs.

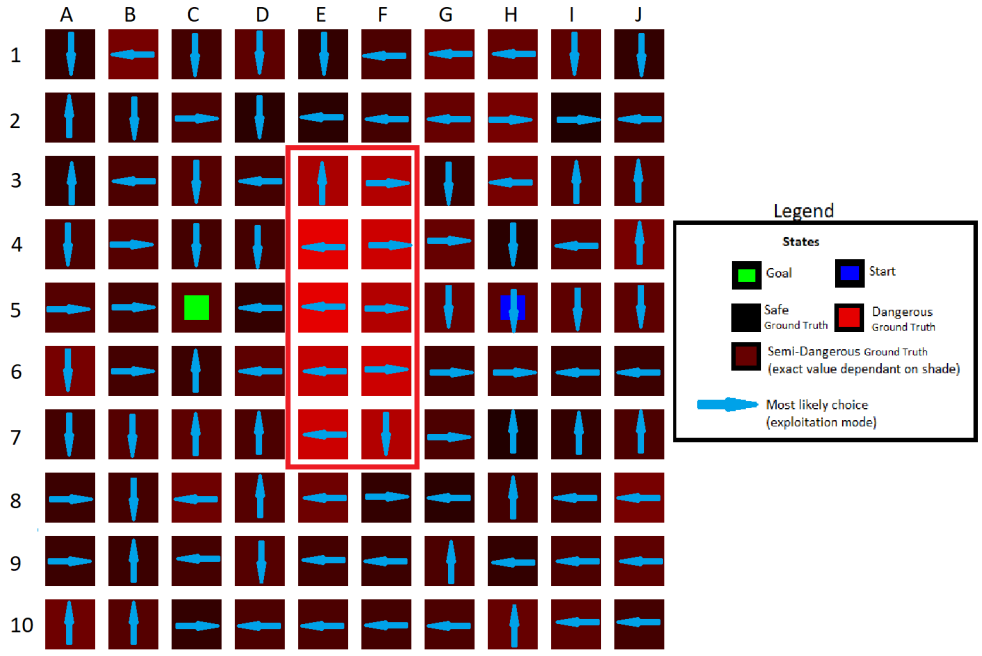

Figure A.10: The most likely action by our system in runtime mode for each square on the danger region gridworld backed the danger value of each square.

The most likely choice in each square is shown in Figure A.9 and A.10. This shows that the system not only avoids the dangerous region as in the previous example but also flows around it more fluidly.

A.2 Basic Quantitative Comparison Experiment - Other Worlds

A simple comparison on the relative successes of both systems on the example gridworlds is shown in Figure A.11 and Table A.1 - our system represented by the red bars and the comparison learner represented by the green bars. These examples are made specifically to show the benefits of our system and so represent greater differences than most real world conditions. First each learner was allowed to explore the gridworld for 10000 runs - our system doing so in **danger training** mode and the comparison learner doing so with parameters set to benefit exploration. Each learner was then made to attempt another 1000 runs of the gridworld - our system doing so in **danger avoidance** mode and the comparison learner doing so with parameters set to benefit exploitation.

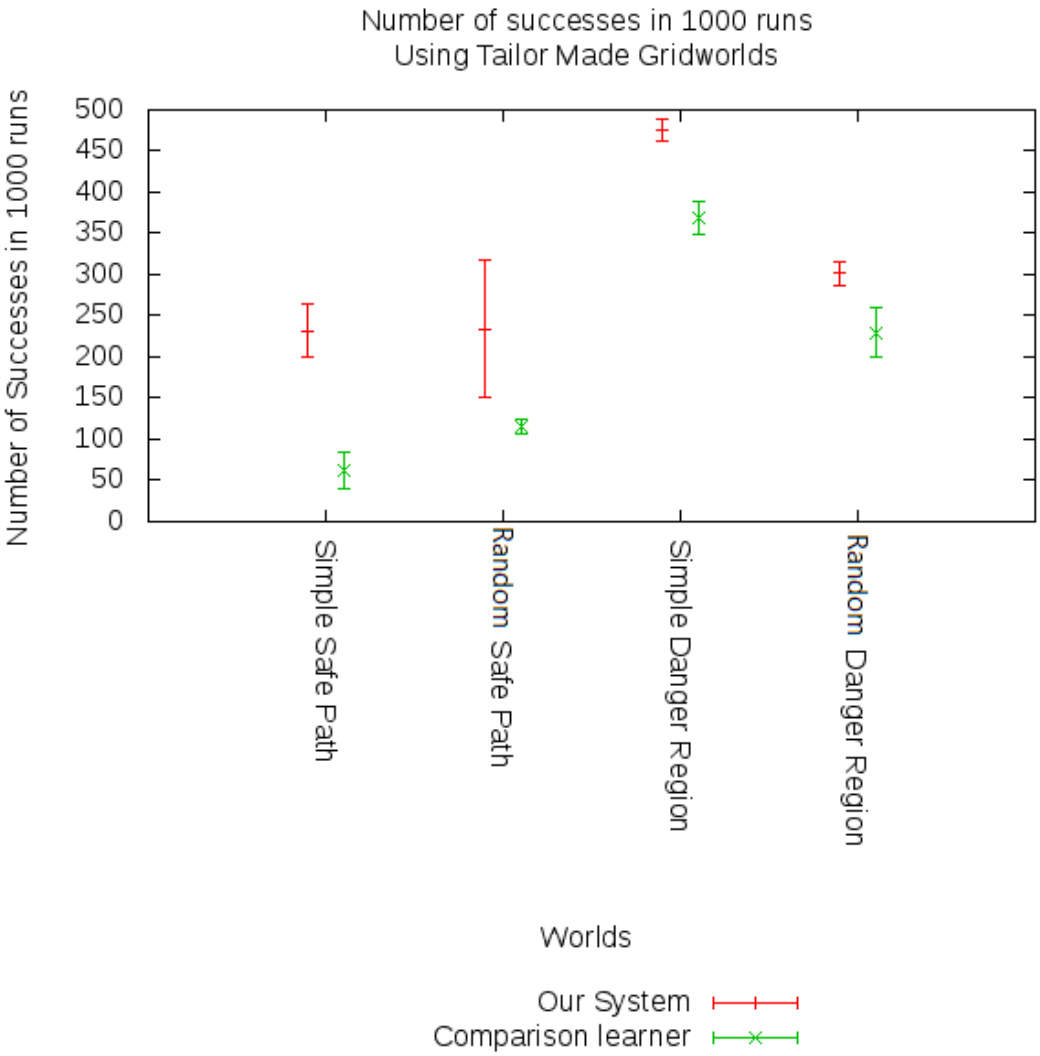

Figure A.11: A basic comparison of the results of the two learners on these example gridworlds. Error bars represent standard deviation - each learner's results offset only for readability. In these examples our systems perform significantly better than the comparison learner but in the safe path examples show significant variance.

World	Our System	Comparison Learner	Straight Line
Random Danger Region	298	262	498
Simple Danger Region	456	358	561
Random Safe Path	256	83	686
Simple Safe Path	179	39	598

Table A.1: A comparison of the performance of both systems given tailor made worlds.

Every time the gridworld moves to a new square it potentially triggers transition to a failure state, the likelihood of which determined by how dangerous the state is. If the agent transitions to a failure state our system begins a new run without making any further actions and the current run is not considered a success. A success is defined as a run in which the agent successfully reaches the goal square without transitioning to a failure state.

In randomly generated examples the goal will not appear with less than two squares separating it from the start location.

The graph shows the number of successful runs each system made in the 1000 tries. While it is clear that the safe path example is more prone to failure than the danger region example as expected our system performs more effectively than the comparison learner. In both of these examples there is a clear safe path or paths and there is a clear dangerous region that an agent will move through if it takes the direct path. As such our system is able to navigate around this obstacle or follow the safe path whereas the comparison system is less likely to do so.

The worlds selected are the examples previously used to generate heatmaps; they are the worlds shown in Figures A.1, A.6, 4.1 and 4.11 respectively.

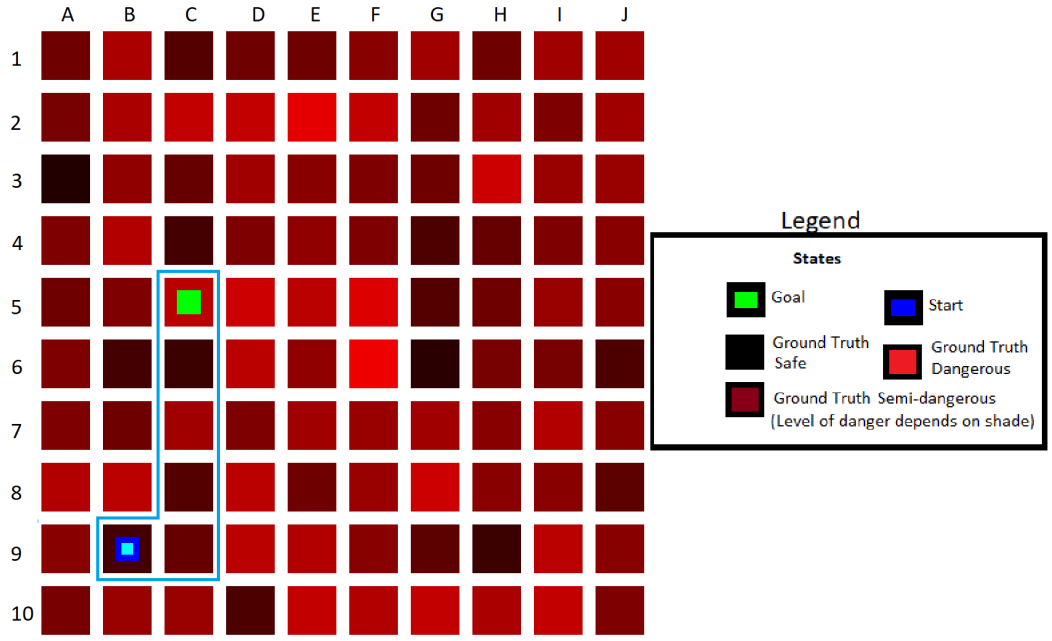

Figure A.12: Random World 1. A relatively safe path is present between the start location and goal which is shown by the blue box.

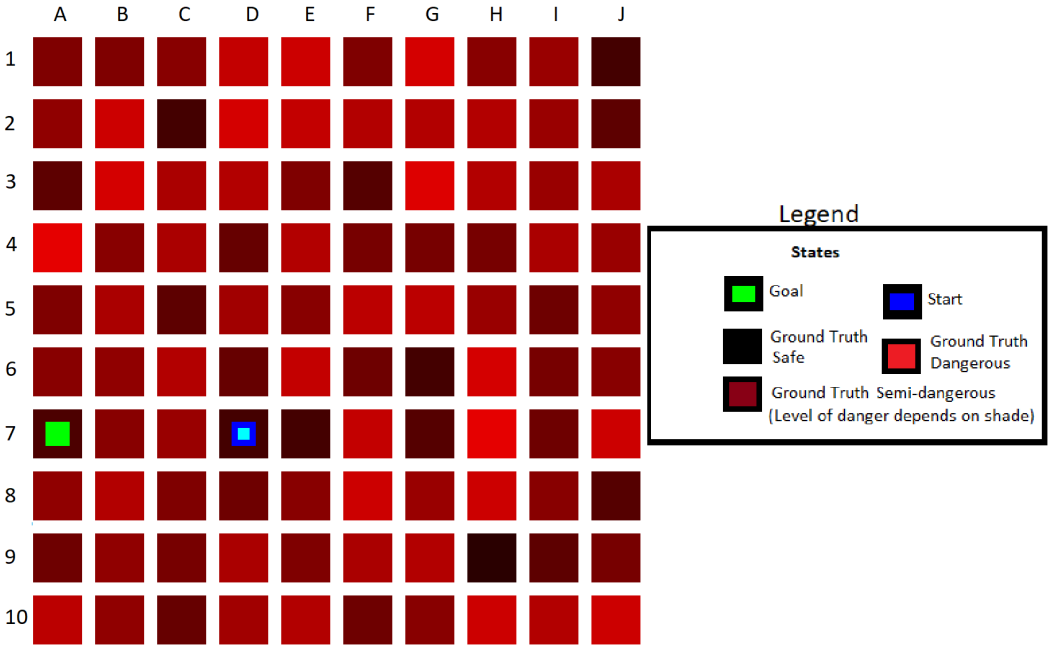

Figure A.13: Random World 2. The goal has generated as close to the goal is as allowed by the rules of the random world generator.

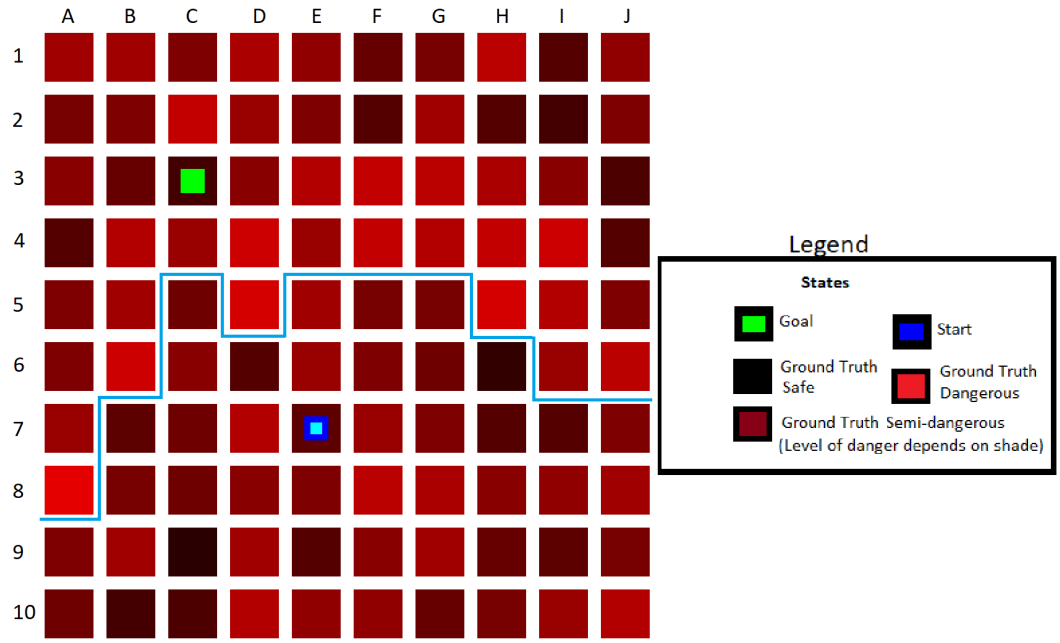

Figure A.14: Random World 3. There is no relatively safe path as a region of high risk states separates the origin and the goal which is shown by the blue line.

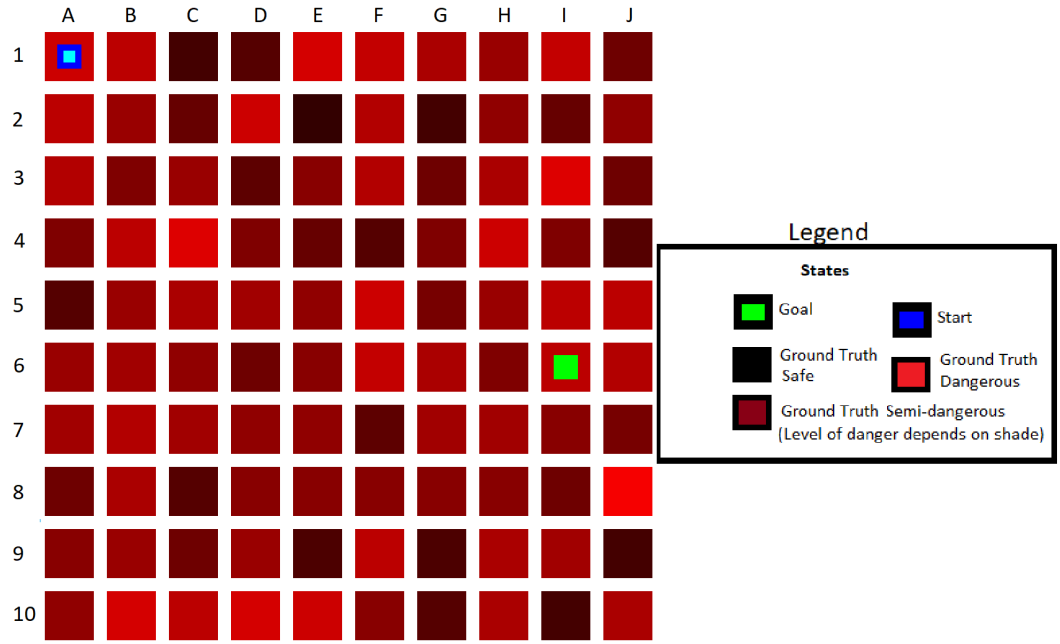

Figure A.15: Random World 4. A fairly uniformly dangerous world.

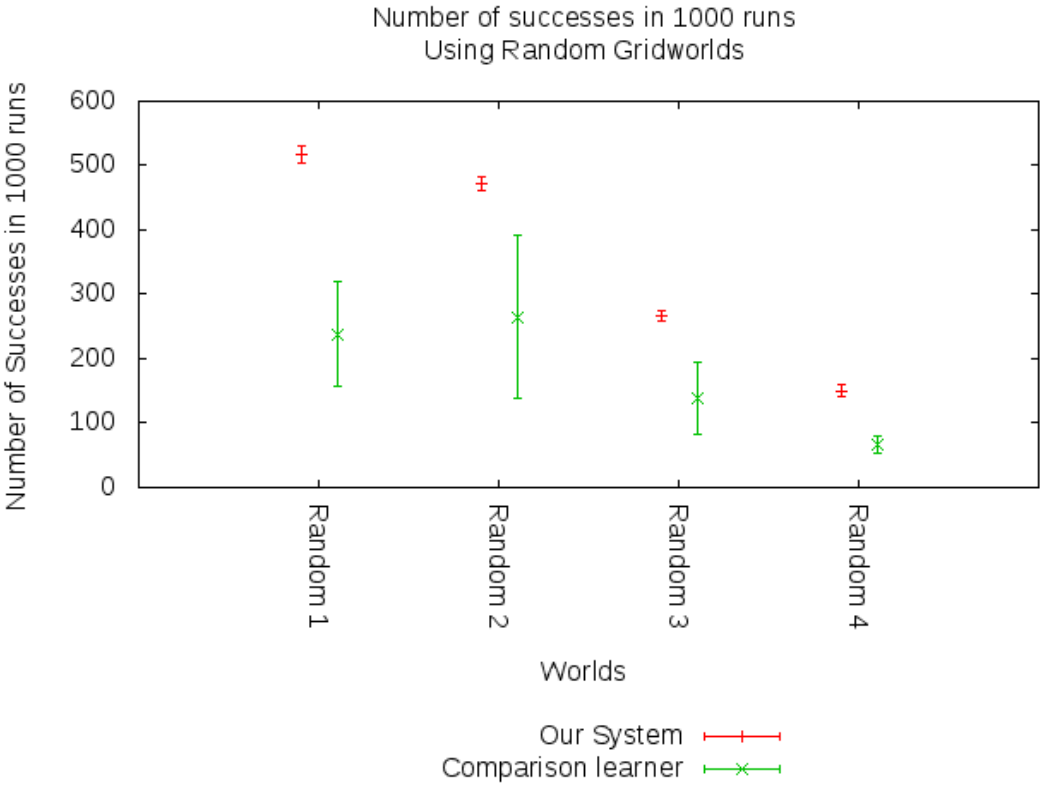

Figure A.16: A basic comparison of the results of the two learners on these example gridworlds. Error bars represent standard deviation - each learner's results offset only for readability. Our system performs much better on these examples with less variance.

World	Our System	Comparison Learner	Straight Line
Random 1	521	263	881
Random 2	352	171	897
Random 3	110	261	806
Random 4	136	16	599

Table A.2: A comparison of the performance of both systems given randomly generated worlds.

A comparison between the performance of our system and the comparison learner in a number of non-tailored examples is shown in Figure A.16 and Table A.2. In the cases shown here each world is randomly generated; the start and finish locations and the danger values of each square being randomly determined

(albeit with a hard limit that the goal cannot be within 2 squares of the start).

The simulation was run the same as in the previous comparison; each learner was allowed to explore the gridworld for 10000 runs in **danger training** mode. Then each system was made to try and reach the goal for 1000 runs in **danger avoidance** mode. Both sets of runs were performed in heuristic mode. The number of times each system was able to reach the goal in each world was recorded and compared.

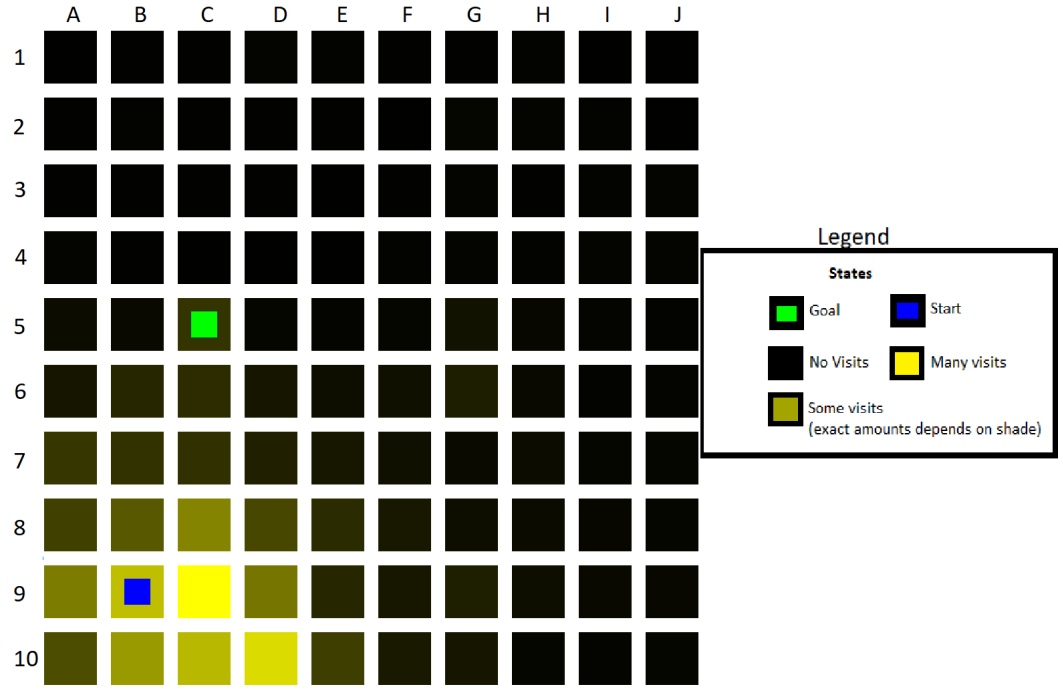

Figure A.17: A heatmap of the number of times each square was visited in 1000 runs using runtime mode in the random world 1 after our system has explored for 10000 runs in training mode. Our system follows a relatively safe path to the goal.

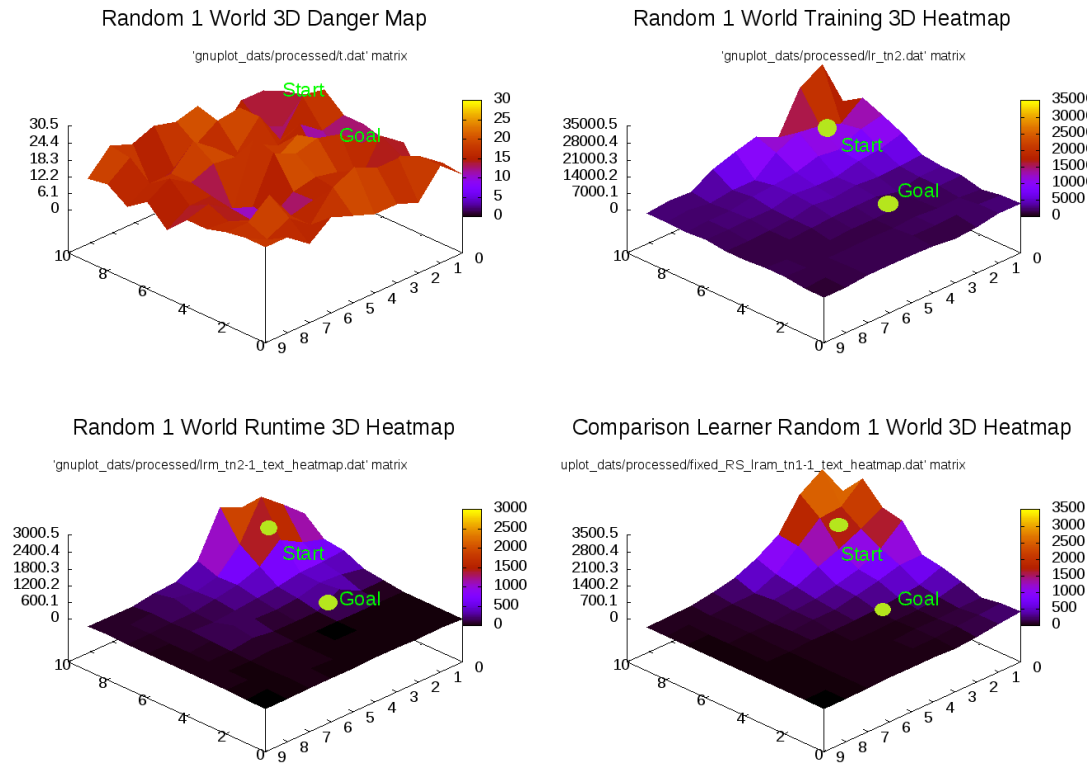

Figure A.18: 3D danger map of the Random 1 World (Upper Left), a 3D heatmap of the 10000 training runs (Upper right) and a 3D heatmap of the 1000 runtime runs of both our system (Lower Left) and the comparison learner (Lower Right).

The most promising result is random world 1 as shown in Figure A.12. In this randomly generated world there is a rough safe path between the start state and the goal. As shown in Figure A.18 in **danger avoidance** mode our system is able avoid the danger encountered by going directly upwards from the start state or from going further to the right and use the ideal path from *C9* to *C5* significantly more often than the comparison learner. A runpath of the learner performing this is shown on Figure A.19.

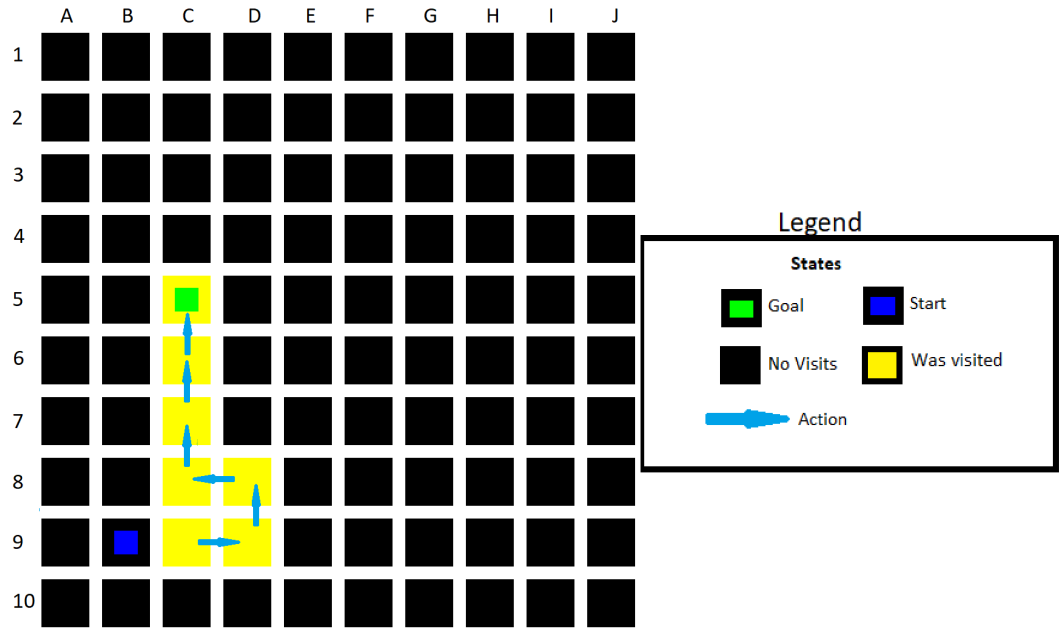

Figure A.19: The runpath of a single example in runtime mode on the random world 1.

In random world 2 as shown in Figure A.13 the start and goal were generated as close to each other as allowed by the hard limit. None the less our system was able to perform slightly better by avoiding a few dangerous squares such as B8 and C6.

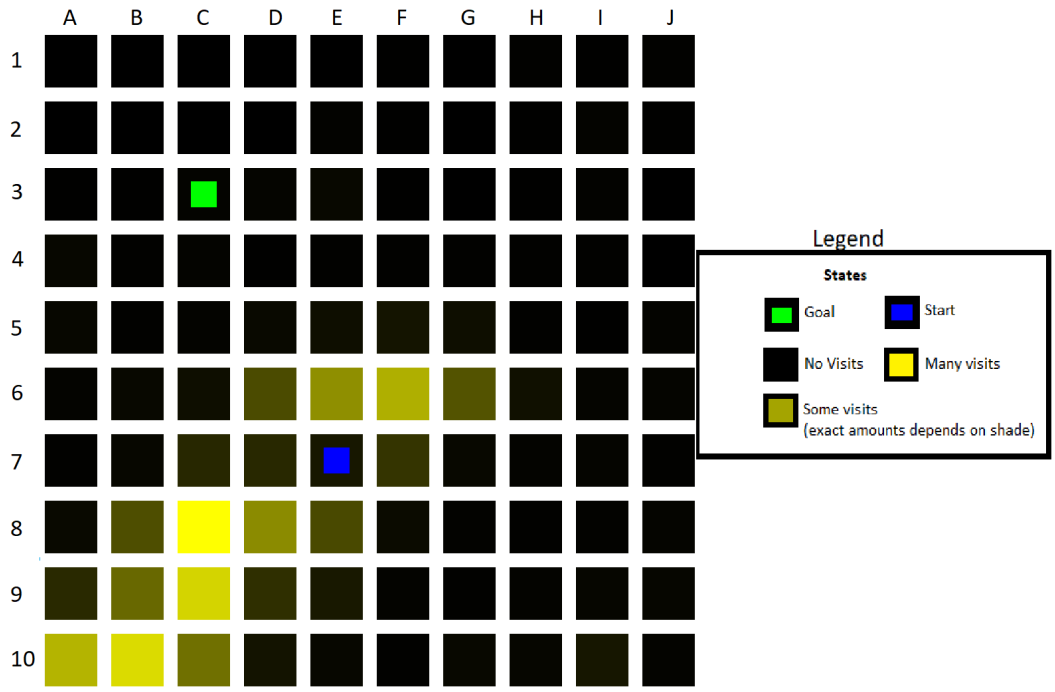

Figure A.20: A heatmap of the number of times each square was visited in 1000 runs using runtime mode with our system in random world 3 after our system has explored for 10000 runs in training mode.

The most disappointing result is Random World 3, as shown in Figure A.14. One possible cause of this reduced success may be the combination of a lack of a safe path between the start and the goal combined with the two false safe paths (a path of safe squares that does not ultimately lead to the goal) to the left and right of the start location; causing the learner to take more actions (and thus encounter more risk) by following these paths but ultimately crossing the high risk region to reach the goal anyway.

This displays one problem our system can encounter; local maxima. The agent is attempting to choose safe actions even if there is no safe path to the goal. If this happens the agent can become trapped in a region of relatively safe space, a local maxima characterized by having a lower probability of failure. If the agent does not transition to a failure state the random element of our system's action selection will make the agent leave the safe area and cross the dangerous region eventually but if the relatively safe region is still somewhat dangerous there is a chance the agent will transition to a failure state before then. This current set up is used for testing purposes to understand how this type of learner operates, relying on randomness to escape the area of low danger is not practical for real

applications as will be discussed in future work.

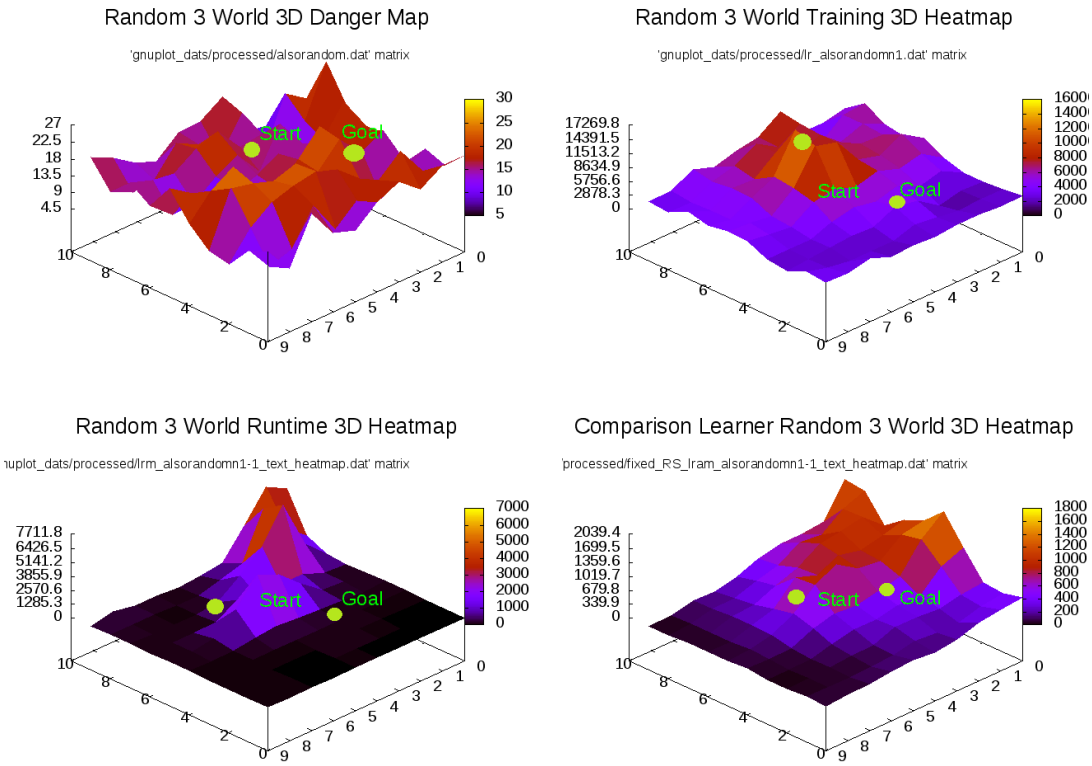

Figure A.21: A 3D danger map of the Random 3 World (Upper Left), a 3D heatmap of the 10000 training runs (Upper Right) and a 3D heatmap of the 1000 runtime runs in both our system (Lower Left) and the comparison learner (Lower Right).

This is shown in the heatmaps shown in Figure A.21. In the **danger avoidance** mode heatmap both paths, particularly the lower path, are overrepresented compared to more straightforward paths towards the goal. The comparison learner on the other hand also moves into this region but is more likely to leave.

A runpath of this example is shown in A.22. At first our system makes the mistake of moving into the lower false path. After spending some time moving around this region it was able to escape and move to the goal.

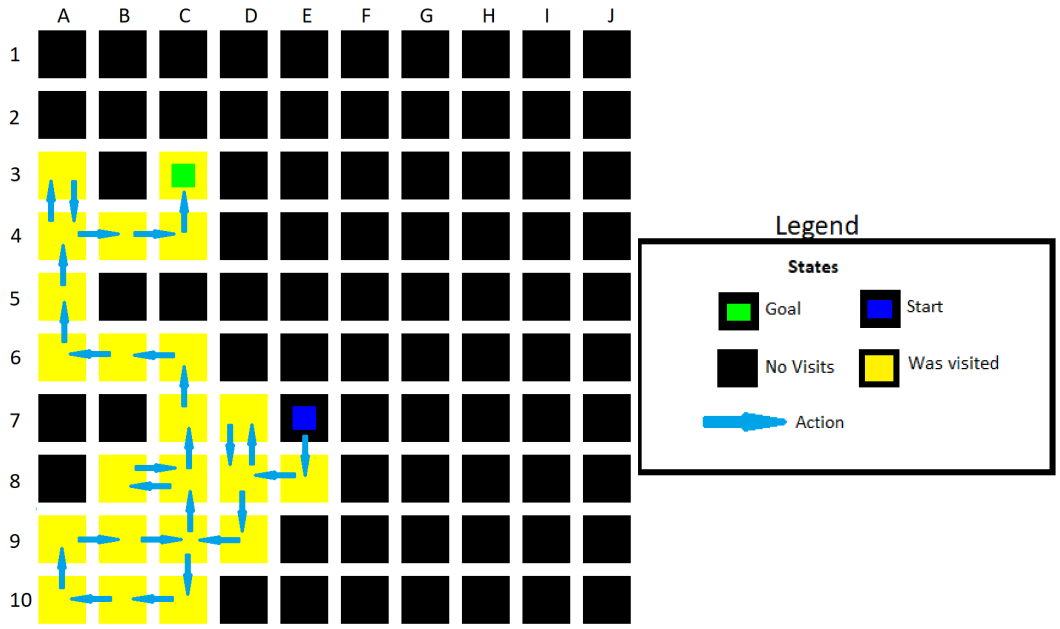

Figure A.22: The runpath of a single example in runtime mode on random world 1.

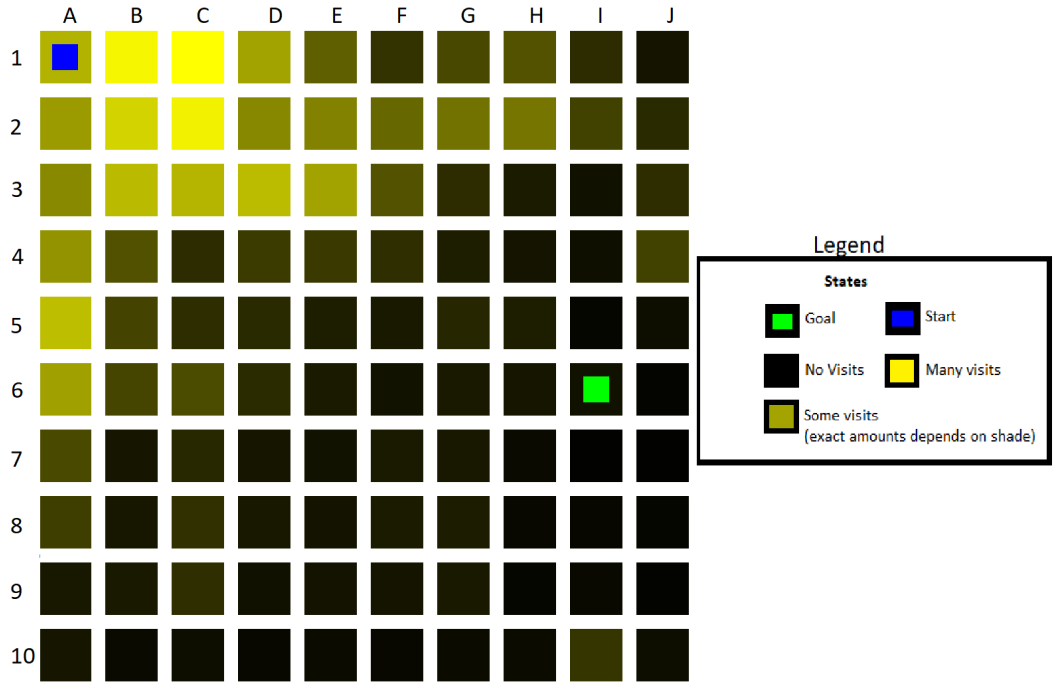

Figure A.23: A heatmap of the number of times each square was visited in 1000 runs using runtime mode with our system in the random world 4 after our system has explored for 10000 runs.

The results of random world 4 as shown in Figure A.15 show what occurs when the space is relatively uniformly and highly dangerous. Our system is able to avoid the worst states giving it a slight advantage but both learners suffer heavy losses in such an environment.

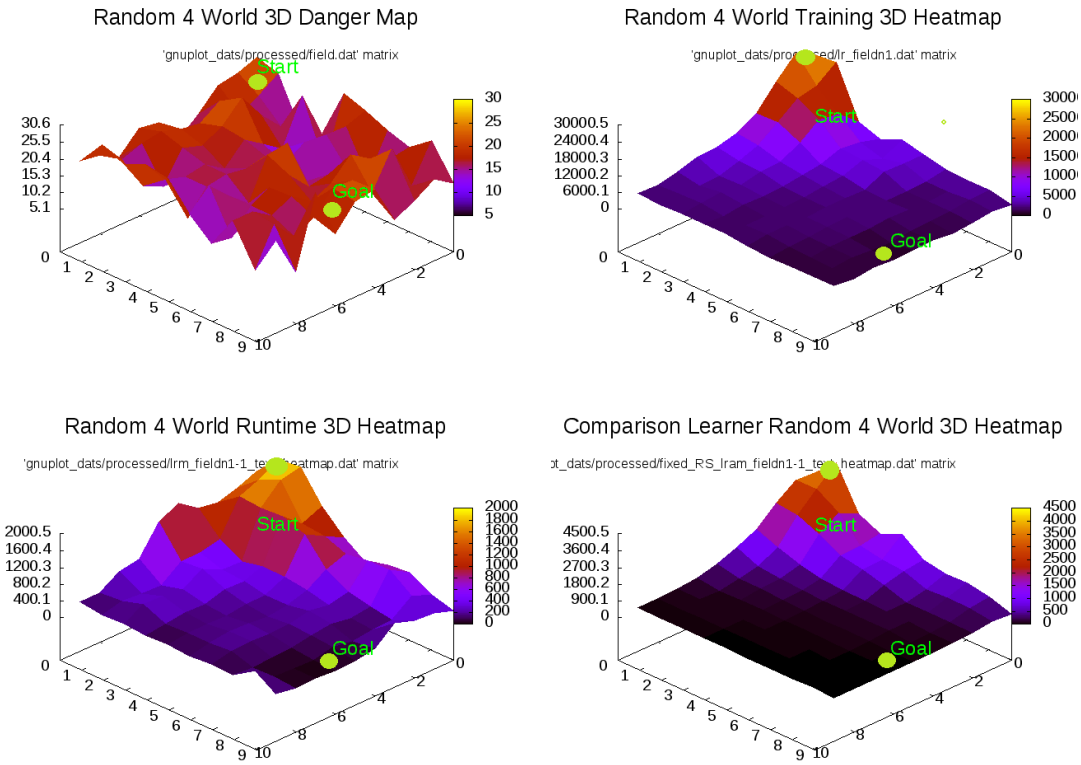

Figure A.24: A 3D danger map of the Random 4 World (Upper Left), a 3D heatmap of the 10000 training runs (Upper Right) and a 3D heatmap of the 1000 runtime runs of both our system (Lower Left) and the comparison learner (Lower Right).

The heatmaps of this world are shown in Figure A.24. This shows the quick drop off caused by the general high danger of the world. This effect is present in all 3 heatmaps as it is a product of the general high danger rate of the world, not the individual strategy of each agent.

Also notable is the presence of some squares such as *I10* which are brighter than all of the squares leading to them. This can be caused when one square is notably safer than all of the surrounding squares. Our system may spend some time in the local maxima of the safer square before breaking out due to the random element of action selection, resulting in multiple visits to the safe

state. As previously stated relying on randomness to escape this area would not be practical in a real application and this will be discussed in future work.

Overall these results show that our system can produce significant advantages over the comparison learner in many scenarios, particularly where there exists a safe path between the start and the goal that allows our system to exploit its advantages, but it is not inherently superior in all scenarios.

Also note the comparison with a straight line goal seeker agent in tables A.1 and A.2. The straight line goal seeker agent first moves left or right until it matches the goal on the x-axis and then moves up or down until it matches the goal on the y-axis - paying no attention to which areas may harbour danger. Notably it achieves results superior to both our system and the comparison learner. However the reason for this is that both systems select their actions probabilistically rather than absolutely - as such both systems will wander to some degree compared to the straight line agent. This example was created to demonstrate the ability of our learner to detect and map low probability dangers but even the safest squares have some probability of triggering a failure state. As such it is advantageous to take as few steps as possible and the benefit of taking fewer steps seems to outweigh the benefit of actually learning the environment.

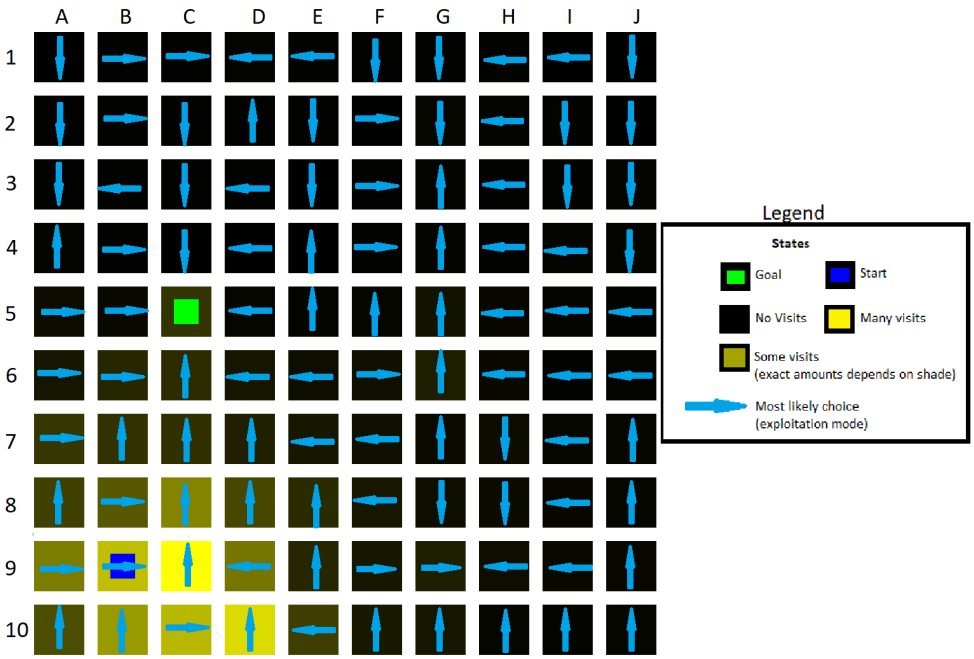

Figure A.25: The most likely action by our system in runtime mode for each square on the Random 1 World backed by a heatmap of 1000 runtime mode runs.

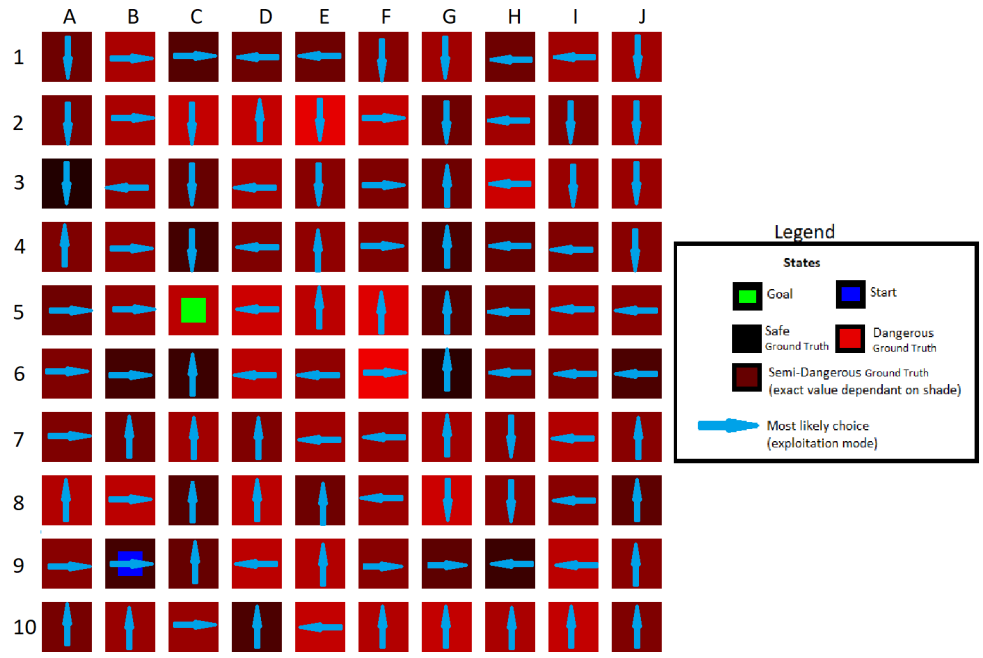

Figure A.26: The most likely action by our system in runtime mode for each square on the Random 1 World backed the danger value of each square.

The most likely choice in each square in Random World 1 is shown in Figure A.25 and A.26. Our system is likely to take a fairly direct path to the goal while avoiding a number of dangerous states.

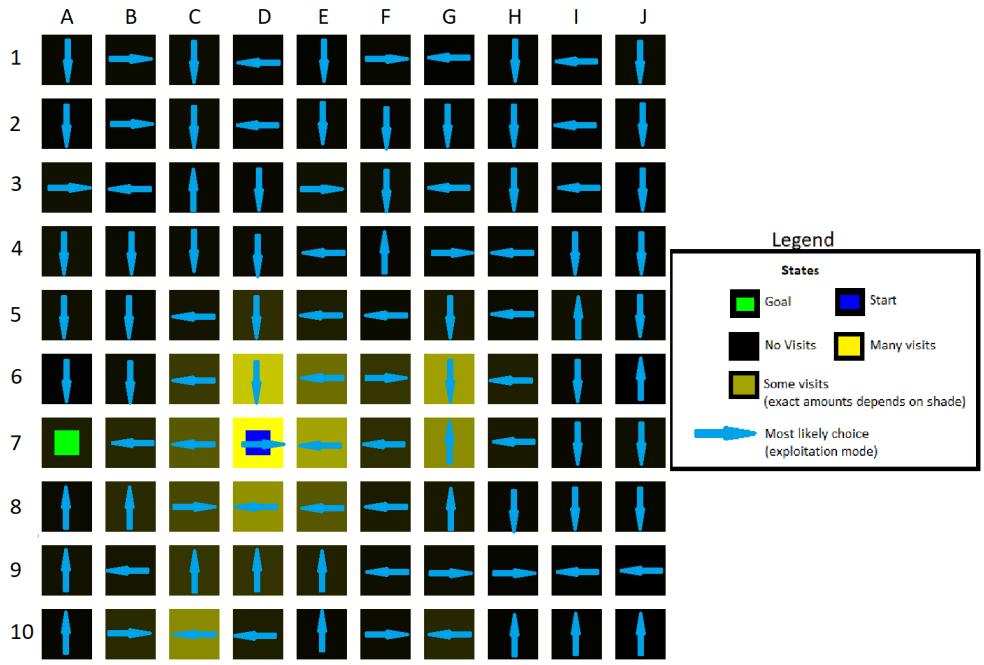

Figure A.27: The most likely action by our system in runtime mode for each square on the Random 2 World backed by a heatmap of 1000 runtime mode runs.

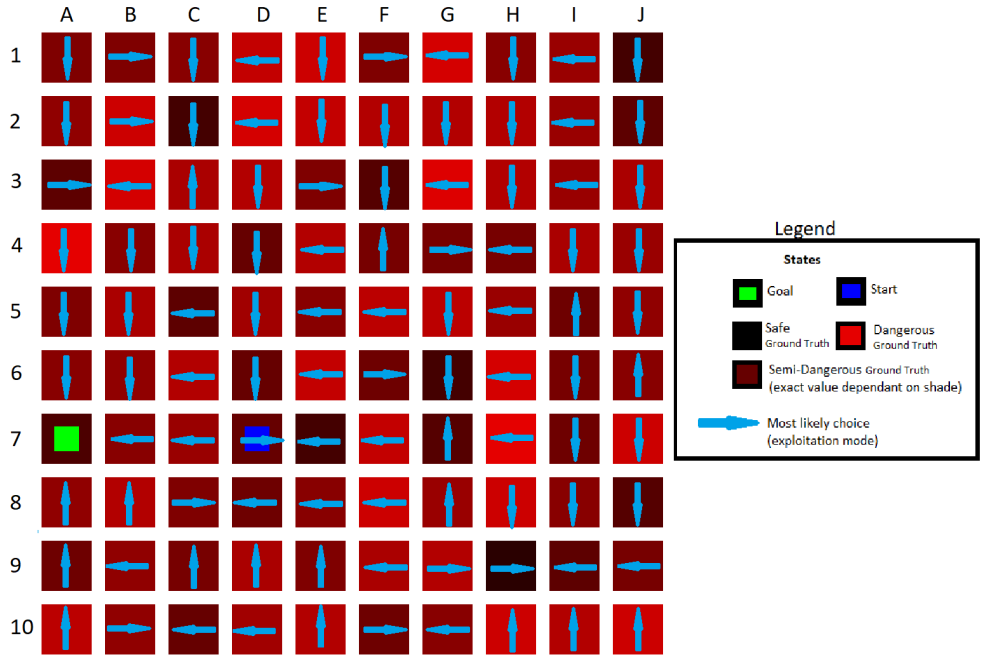

Figure A.28: The most likely action by our system in runtime mode for each square on the Random 2 World backed the danger value of each square.

The most likely choice in each square in Random World 2 is shown in Figure A.27 and A.28. While most of the most likely moves are productive our system will likely move in the wrong direction for the first step.

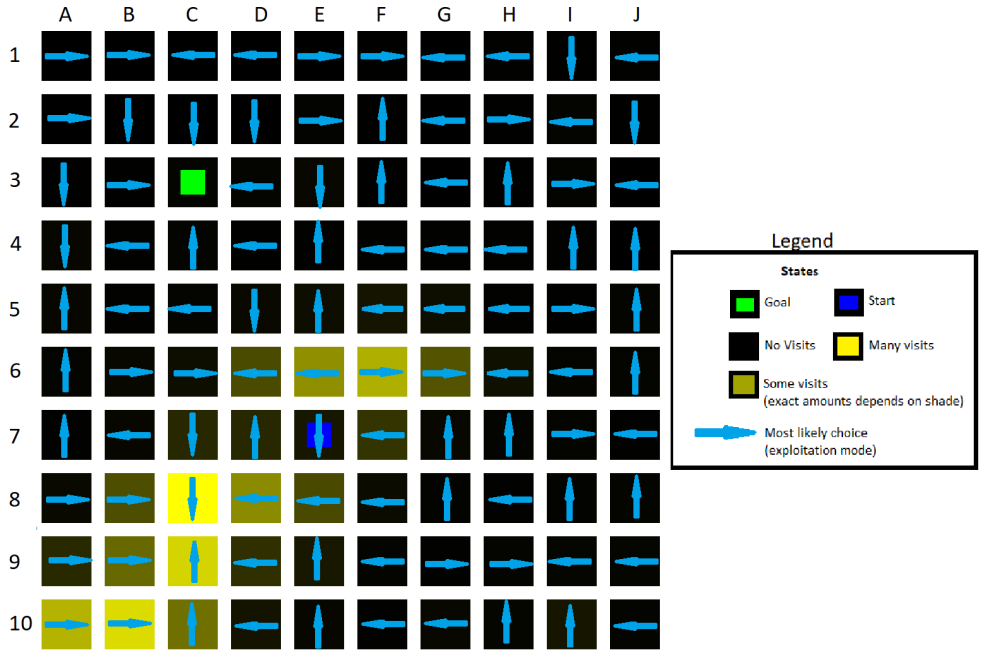

Figure A.29: The most likely action by our system in runtime mode for each square on the Random 3 World backed by a heatmap of 1000 runtime mode runs.

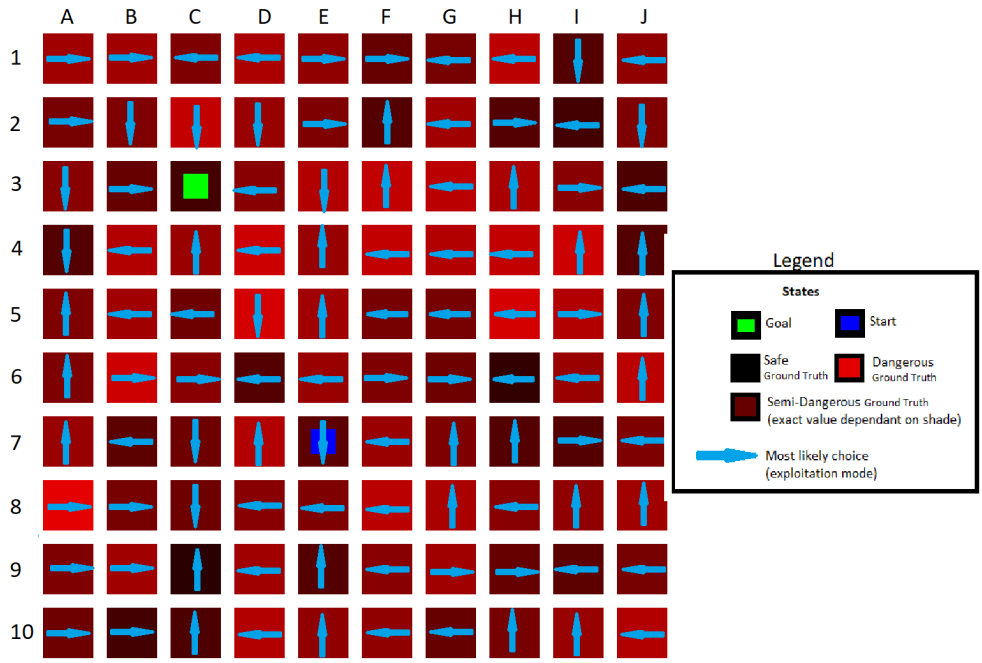

Figure A.30: The most likely action by our system in runtime mode for each square on the Random 3 World backed the danger value of each square.

The most likely choice in each square in Random World 3 is shown in Figures A.29 and A.30. The most likely actions may keep the agent trapped in the lower left corner.

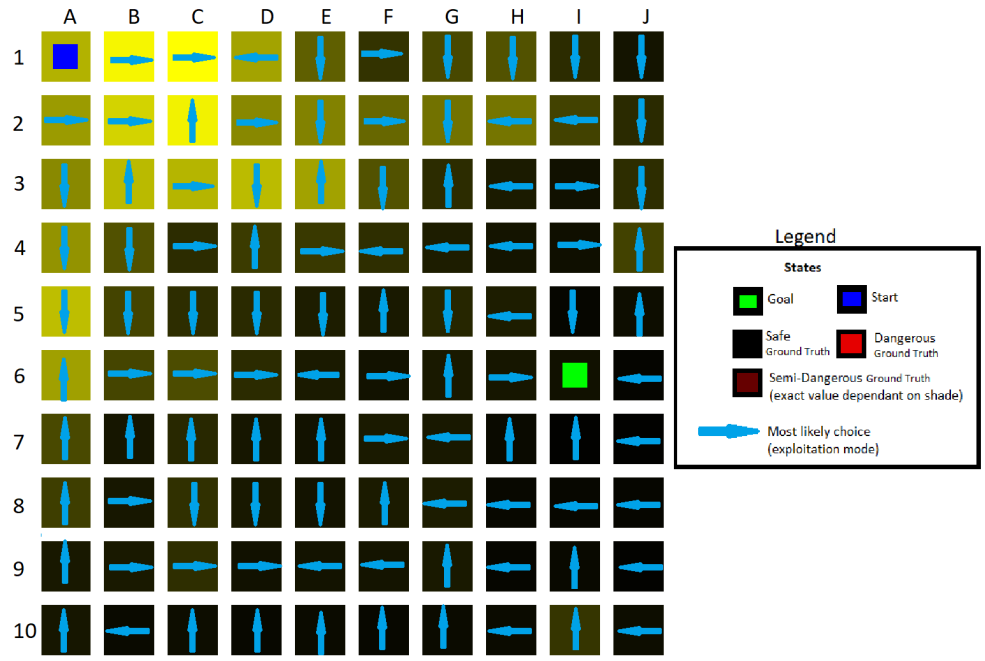

Figure A.31: The most likely action by our system in runtime mode for each square on the Random 4 World backed by a heatmap of 1000 runtime mode runs.

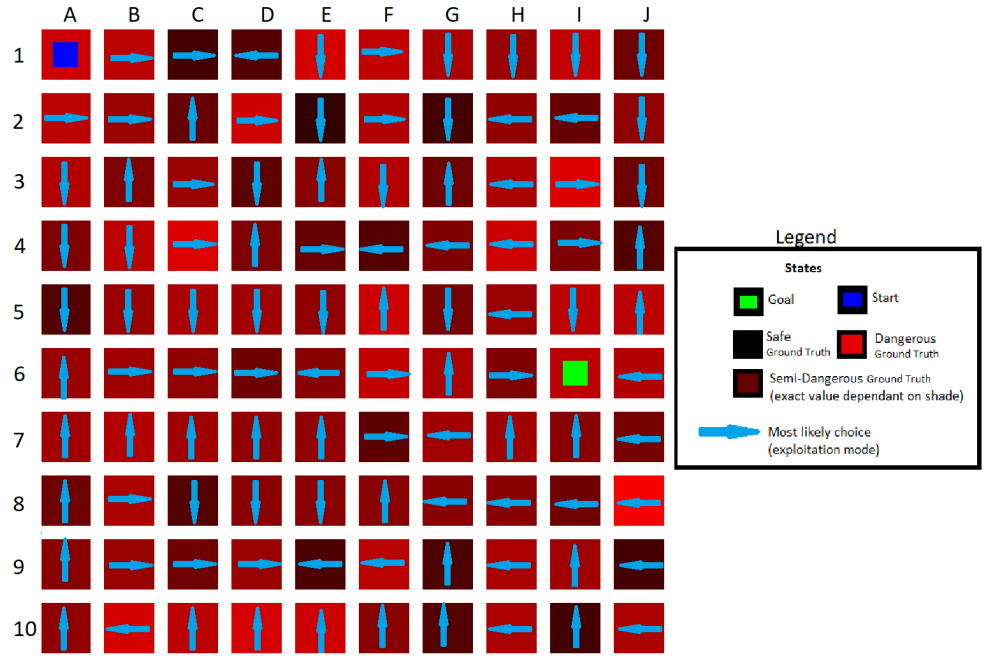

Figure A.32: The most likely action by our system in runtime mode for each square on the Random 4 World backed the danger value of each square.

The most likely choice in each square in Random World 4 is shown in Figures A.31 and A.32. Most of the most likely moves are in sensible directions and attempt to avoid the most squares where possible. This shows that our system is making sensible decisions but the extremely high average danger of this world still results in a low success rate.

A.3 Dangerous Square Density Experiment

We will examine the effect of the number of dangerous squares on the performance of both our system and the comparison learner.

Nine semi-random worlds were generated - in each world the number of dangerous squares was fixed to a particular percentage. The location of these dangerous squares as well as the location of the start state and goal however were randomized. Each dangerous square is maximally dangerous and each safe square is minimally dangerous.

Both learners were allowed to explore every world for 1000 runs in **danger training** mode in the case of our system and with parameters set to benefit exploration in the case of the comparison system. Our learner performed these runs in heuristic mode.

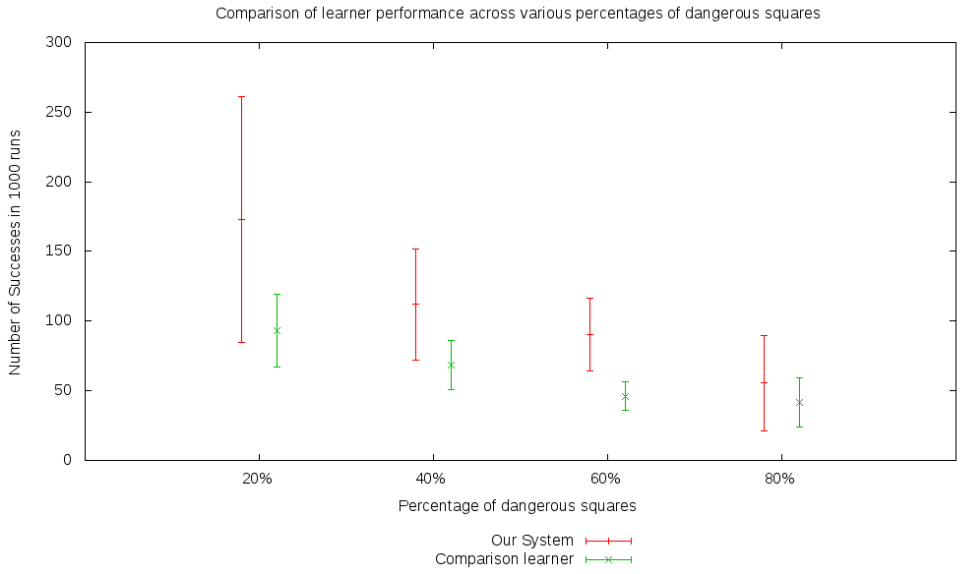

Figure A.33: The average success rate per 1000 runs for the fixed danger percentage worlds given limited training data. Based on 100 runs with 100 separate worlds.

Percentage of dangerous squares	Our System Mean Successes	Our System Std Dev	Comparison Learner Mean Successes	Comparison Learner Std Dev
20%	172.8	88.3	93.1	26.01
40%	112.0	40.0	68.4	17.91
60%	90.4	26.21	46.2	10.19
80%	55.5	34.19	41.8	17.7

Table A.3: The average success rate per 1000 runs for the fixed danger percentage worlds given limited training data. Based on 100 runs with 100 separate worlds.

The results of this experiment are shown in Figure A.33 and Table A.3. The large variance between the runs shows the impact of the position of dangerous states on our system - a safe path can be present in an example with more dangerous states and absent from one with less. Additionally worlds with closer start and goal locations will have more successes due to the reduced quantity of actions, and thus opportunities to transition to a failure state, needed to reach the goal.

We also performed an experiment on the effect of how close the start and goal locations are on the final results. In order to test this we generated a number of worlds that are identical aside from the distance between the start and goal locations.

Five worlds were generated with a distances of 12, 10, 8, 6 and 4 respectively Distance is measured as Manhattan distance as the agents are incapable of moving diagonally within the gridworld.

As with previous examples the learners were allowed to navigate the world for 10000 runs in **danger training** mode prior to a further 1000 runs in **danger avoidance** mode. We will compare the number of successful **danger avoidance** mode runs in at each distance with both learners.

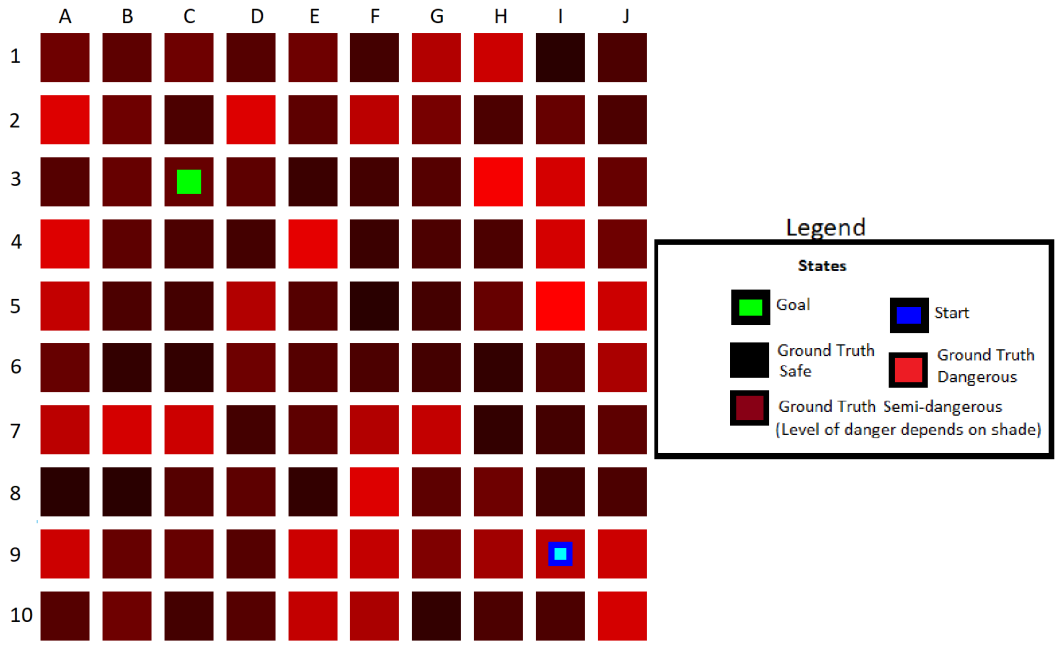

Figure A.34: The distance 12 world. The other distance worlds are the same except for start and goal locations.

Distance	Start	Goal
12	I9	C3
10	H8	C3
8	G7	C3
6	G7	D4
4	F6	D4

Table A.4: The locations of the start and goal locations in each of the example worlds.

The danger landscape present in all of these worlds is shown in Figure A.34, the start and goal locations of each world are shown in table A.4.

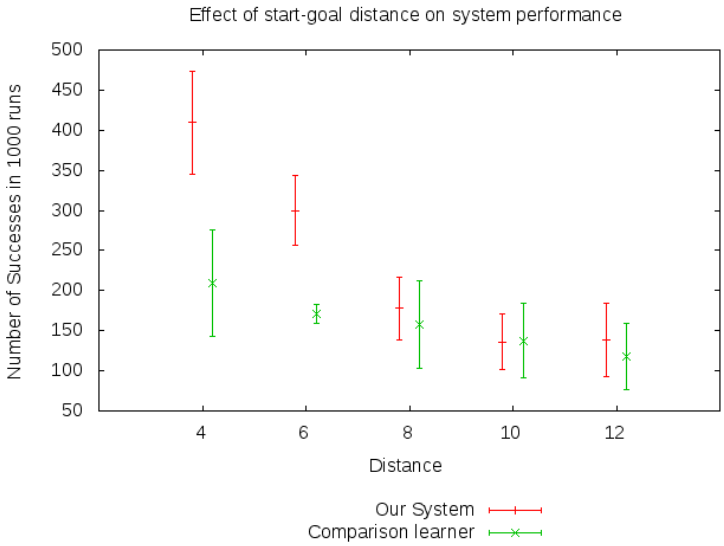

Figure A.35: A comparison of the results of altering the distance between the start and the goal. Error bars represent standard deviation.

Distance	Our System Mean Successes	Our System Std Dev	Comparison Learner Mean Successes	Comparison Learner Std Dev
4	409.6	64.15	209.6	66.69
6	199.7	43.24	170.8	11.31
8	177.7	39.22	157.8	54.87
10	136.3	34.88	137.5	46.05
12	138.6	46.16	117.4	41.39

Table A.5: The mean number and standard deviation of successful runs out of 1000 each learner performed at each distance.

The results of this experiment are shown in Figure A.35 and Table A.5. As expected, in general decreasing the distance between start and the goal increases the success rate of our system.

The comparison learner shows on the other hand does not show consistently improved performance. This may be due to the presence of the two dangerous squares in *E4* and *D5*. These squares are close to the goal and in the direct path between the start and the goal. This shows that while decreasing the distance between the start and the goal will generally be beneficial it will not always do so depending on the arrangement of dangerous squares in the gridworld.

A.4 Effect of training on performance

In this experiment we will examine the effect of the number of training runs on the performance of our system and determine the point at which additional training runs will not improve system performance.

The number of prior **danger training** runs will be varied while the number of *danger avoidance* mode exploitation runs will remain constant at 1000. These tests were performed on the Random World 2 shown in Figure A.13 with our learner operating in heuristic mode. This world was chosen because it is a fairly normal random world, not too safe but not too dangerous.

The number of successful exploitation runs in each test will be compared as a metric to examine the performance of our system.

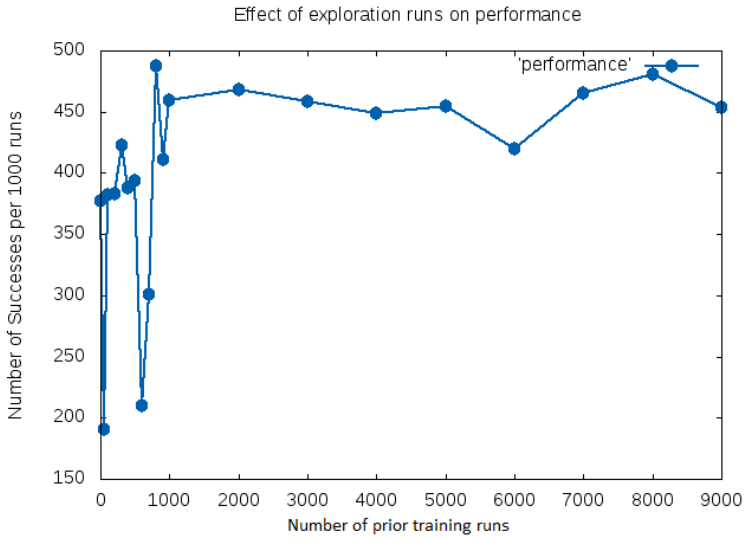

Figure A.36: A comparison of the performance of our learner depending on the number of training runs it is allowed to perform.

Num of exploration runs	Num of successes	Num of exploration runs	Num of successes
50	191	1000	459
100	382	2000	468
200	383	3000	458
300	423	4000	449
400	388	5000	455
500	394	6000	420
600	210	7000	465
700	301	8000	481
800	487	9000	454
900	411		

Table A.6: A comparison of the performance of our learner depending on the number of training runs it is allowed to perform. Grey entries left intentionally blank.

The results of this experiment are shown in Figure A.36 and Table A.6. Performance stabilises at around 1000 **danger training** runs at approximately 450 successes per 1000 **danger avoidance** mode exploitation runs. Prior to this our system gains a proficiency of around 350 successful exploitation runs within 100 exploitation runs, albeit with a degree of inconsistency which sometimes leads our system to substandard results.

This shows that our system ceases to improve with more data beyond a certain point. This is likely due to two main potential issues:

- The first is the problem of local minima previously explained in Section 4.5 in which our system will be drawn to relatively safe locations in the environment even if it is necessary to cross dangerous regions in order to reach the goal.

- The second is the problem explained in Section A.3; our system chooses the safest options even if all options are relatively dangerous. However in a uniformly dangerous environment such as Random World 4 shown in Section 4.5 a more direct path to the goal can be safer overall as it reduces the number of actions and thus failure state risks overall the agent is exposed to; the total risk of these actions can be lower even if each individual risk is higher. This problem may be solved by modifying our system to estimate

discounted danger as opposed to immediate danger but this remains future work.

Our system is closest to optimal in environments in which a relatively safe path to the goal is present if indirect or difficult for the agent to discover and in which the average risk of the relatively safer states is low enough to avoid penalising the agent for taking addition actions. In these environments our system is capable of outperforming the comparison learner with the quantities of training data used for these experiments.

A.5 Danger Square Density with Limited Data Experiment

Set Up

In this experiment we will repeat the experiment performed in Section A.3 in which a number of runs are performed on worlds with different percentages of dangerous squares but drastically limit the number of runs in both the training and exploitation phase.

Aims

This will allow us to compare effect of danger density on low exploration runs for both our system and the comparison learner.

Data Generation

In our experiment we will allow our system and the comparison learner to navigate each of the worlds used in the experiment described in Section A.3 for only 50 runs in **danger training** mode before conducting a further 50 runs in **danger avoidance** mode and measuring the number of successes.

In the second experiment we will generate 1000 new worlds at a number of danger levels. In each of these worlds each learner will perform 50 exploration runs and then 1000 runtime runs and the average number of successes measured.

In both of these experiments the system will operate in heuristic mode.

Results

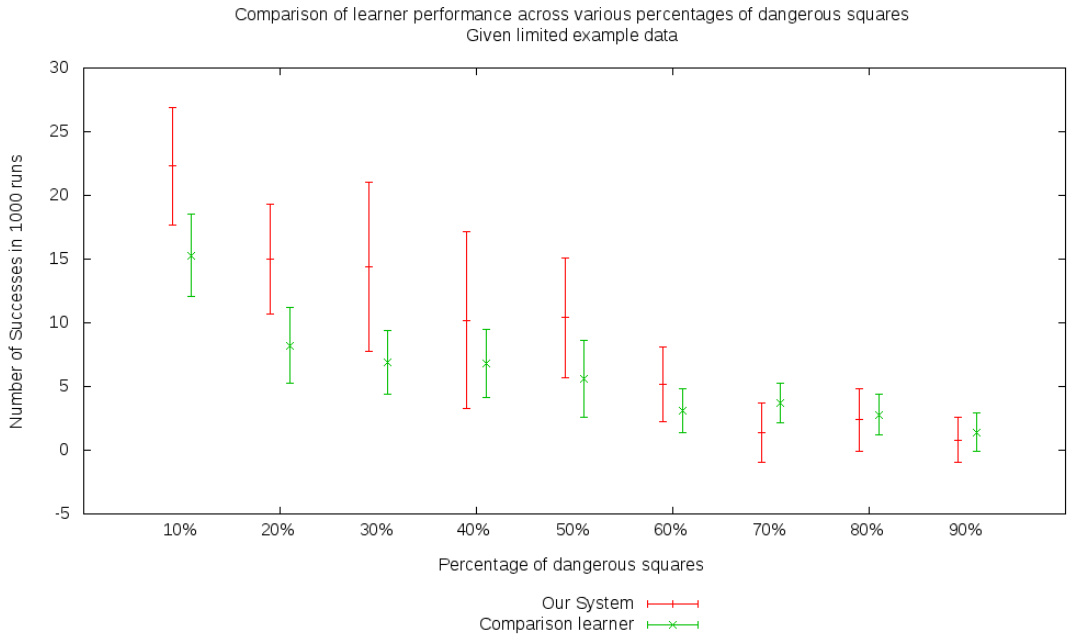

Figure A.37: The average success rate per 50 runs for the fixed danger percentage worlds given limited training data. Error bars represent standard deviation - each learner's results offset only for readability. Our system is able to find safe paths in the low danger worlds but loses its advantage in the high danger worlds.

Percentage of dangerous squares	Our System Mean Successes	Our System Std Dev	Comparison Learner Mean Successes	Comparison Learner Std Dev
10%	22.3	4.62	15.3	3.27
20%	15	4.35	8.2	2.97
30%	14.4	6.64	6.9	2.51
40%	10.2	6.93	6.8	2.7
50%	10.4	4.7	5.6	2.99
60%	5.2	2.94	3.1	1.73
70%	1.4	2.31	3.7	1.57
80%	2.4	2.46	2.8	1.62
90%	0.8	1.75	1.4	1.5

Table A.7: The average success rate per 50 runs for the fixed danger percentage worlds given limited training data.

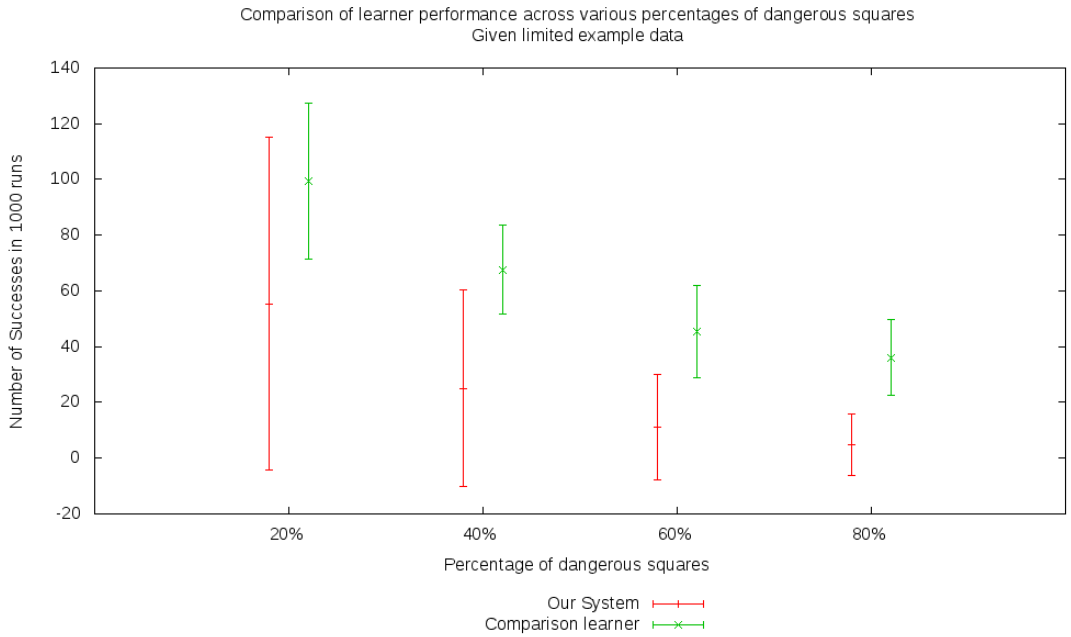

Figure A.38: The average number of successes per 1000 runs in 1000 different worlds (one run per world) after the learners are trained with 50 runs in each world. Error bars represent standard deviation - each learner's results offset only for readability.

Percentage of dangerous squares	Our System Mean Successes	Our System Std Dev	Comparison Learner Mean Successes	Comparison Learner Std Dev
20%	55.36	59.65	99.38	28.1
40%	25.0	35.33	67.5	15.96
60%	11.12	18.75	45.55	16.59
80%	4.68	11.07	36.14	13.74

Table A.8: The average number of successes per 1000 runs in 1000 different worlds (one run per world) after the learners are trained with 50 runs in each world.

We generated 100 worlds with each percentage of dangerous squares. In each of these worlds we trained both our system and the comparison learner with 50 runs in **danger training** mode. The policy was then fixed and a further 1000 runs performed in **danger avoidance** mode and the average number of successes determined. The results for this experiment are shown in Figure A.37 and Table A.7. Our system is generally more effective compared to the comparison learner

when the danger density is relatively low. Lower density increases the chances of an indirect safe path that can be found and decreases the chances of local minima.

Next we repeated this with 1000 different worlds in Figure A.38 and Table A.8. These results are generally less promising for our system but show the same higher variance for our system compared to the comparison learner.

The higher variance for our system as compared to the comparison learner shows that our system is more sensitive to configuration of the environment than the comparison learner which is primarily just affected by the density of dangerous squares. Our system can both more easily find a twisty but safe path to the goal and more easily becomes trapped in the local maxima.

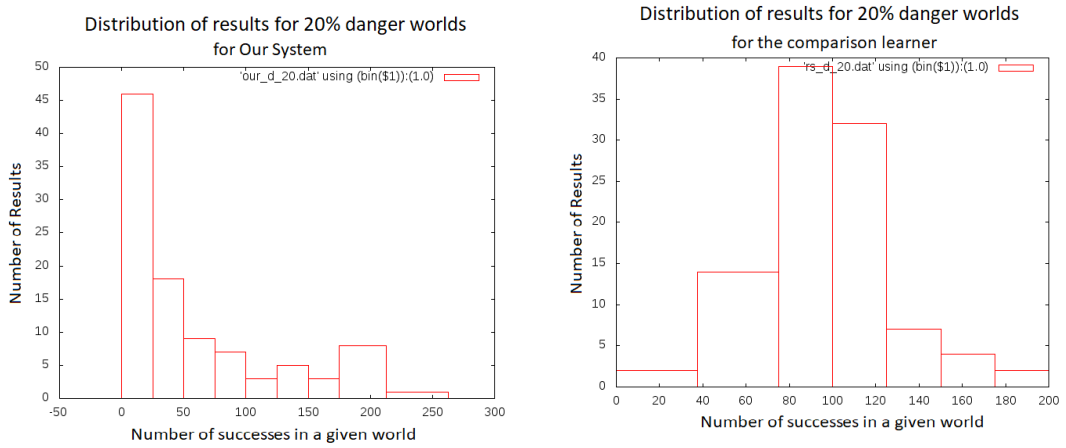

Figure A.39: The distribution of results for both our system (Left) and the comparison learner (Right) on the 20% danger example worlds.

The distribution of results for the 20% danger worlds is shown in Figure A.39. This shows that while, in general, the comparison learner outperforms our system under low training data conditions on randomly generated worlds in which 20% of the squares are dangerous our system achieves 200 successful runs per world more often than the comparison learner. This shows there is a class of problems for which our system is superior even in low training data conditions.

Determining the exact conditions in which this approach is superior remains future work but it is likely superior in cases where there is a relatively safe path to the goal and no local maxima for the agent to become trapped in. Furthermore in low training data examples the complexity of the safe path is likely limited.

A.6 Effect of Relative Weights on Exploration

Set Up

In this experiment we alter the relative weights of $Q(s, a)$ and $D(s, a)$ during the training phase with the system in **danger avoidance** mode in the Simple Safe Path world.

Aims

This will allow us to determine to what extent it is better to explore with a focus on determining the best path compared to a focus on exploring potential risks.

Data Generation

In this experiment we will allow our system and the comparison learner to navigate each of the worlds used in the experiment described in Section A.3 for only 50 runs in **danger training** mode before conducting a further 50 runs in **danger avoidance** mode and measuring the number of successes. Our learner will operate in heuristic mode to reduce the effects of Q-Learning on the final results, this will result in the success rate being lower than if these runs were performed in secondary learner mode.

Results

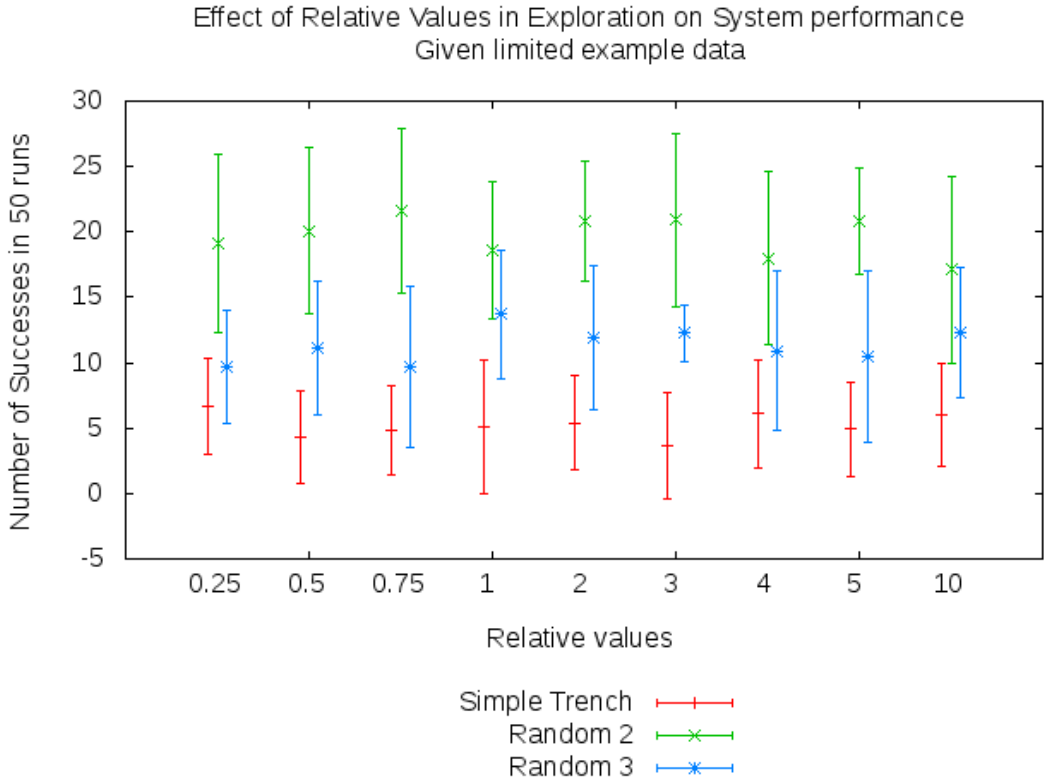

Figure A.40: The average success rate per 50 runs depending on the relative value between seeking the goal and avoiding risks in exploration mode given limited exploration runs. Error bars represent standard deviation - each learner's results offset only for readability. There does not appear to be a consistent improvement of altering the relative values in either direction.

Relative weights	Random 2	Simple Safe Path	Random 3
0.25	19.1	6.7	9.7
0.5	20.1	4.3	11.1
0.75	21.6	4.8	9.7
1	18.6	5.1	13.7
2	20.8	5.4	11.9
3	20.9	3.7	12.3
4	18	6.1	10.9
5	20.8	4.9	10.5
10	17.1	6	12.3

Table A.9: The average success rate per 50 runs depending on the relative value between seeking the goal and avoiding risks in exploration mode given limited exploration runs.

The results of this experiment are shown in Figure A.40 and Table A.9.

In most of these examples the distributions are fairly even, showing decent performance with all weightings.

This suggests that even in examples where one of these quantities is heavily weighted over the other the incidental collection of data about the other value is enough to allow our system to perform adequately.

However in each of the worlds tested the maximum values are either at a relative weight of around one or an even distribution, with the exact value depending on the environment

This suggests shows that taking both $Q(s, a)$ and $D(s, a)$ into account in the exploration maximizes our system's chances of successfully producing a useful model of the environment when given limited exploratory runs.

Appendix B

Numerical Heatmap Tables

B.1 Simple Danger Region Tables

	A	B	C	D	E	F	G	H	I	J
1	507	516	662	840	952	1175	1341	1488	1546	1517
2	468	521	666	728	966	1191	1459	1598	1662	1687
3	437	431	529	575	612	874	1593	1982	2037	1931
4	422	321	331	465	468	872	1913	2429	2264	2207
5	329	239	*456*	369	441	930	2058	**2391**	2564	2160
6	357	285	347	496	507	892	1798	2353	2374	2191
7	469	408	471	608	548	778	1492	1770	1832	1747
8	415	460	543	699	758	929	1310	1512	1636	1632
9	488	590	706	798	905	1030	1276	1319	1425	1379
10	501	624	828	866	984	1001	1260	1314	1356	1269

Table B.1: The exact number of times each square was visited in 1000 runs of the simple wall example by our system in runtime mode. The bolded number is the goal location, the italicised number is the goal location.

	A	B	C	D	E	F	G	H	I	J
1	344	430	474	605	697	799	1015	1189	1292	1264
2	348	442	489	597	711	824	1087	1307	1360	1413
3	366	396	431	555	680	909	1200	1477	1560	1686
4	356	342	329	498	678	1069	1512	2014	1966	1958
5	336	277	*358*	459	750	1259	1939	**2101**	2290	2119
6	416	389	379	646	853	1273	1765	2257	2284	2176
7	520	566	583	770	950	1355	1633	1976	2188	2186
8	684	790	778	945	1040	1427	1729	2001	2086	2143
9	897	886	903	1116	1151	1491	1844	2053	2047	2075
10	941	938	1000	1196	1377	1632	1875	2107	2127	2310

Table B.2: The exact number of times each square was visited in 1000 runs of the simple danger region example by the comparison learner in runtime mode.

	A	B	C	D	E	F	G	H	I	J
1	163	86	188	235	255	376	326	299	254	253
2	120	79	177	131	255	367	372	291	302	274
3	71	35	98	20	-68	-35	393	505	477	245
4	66	-21	2	-33	-210	-197	401	415	298	249
5	-7	-38	*98*	-90	-309	-329	119	**290**	274	41
6	-59	-104	-32	-150	-346	-381	33	96	90	15
7	-51	-158	-112	-162	-402	-577	-141	-206	-356	-439
8	-269	-330	-235	-246	-282	-498	-419	-489	-450	-511
9	-409	-296	-197	-318	-246	-461	-568	-734	-622	-696
10	-440	-314	-172	-330	-393	-631	-615	-793	-771	-1041

Table B.3: The difference in the exact number of times each square was visited in 1000 runs of the simple danger region example by our system and the comparison learner in runtime mode.

B.2 Random Danger Region Tables

	A	B	C	D	E	F	G	H	I	J
1	381	313	381	472	666	842	829	883	989	973
2	348	310	405	503	652	761	888	1069	1351	1196
3	282	274	331	400	493	642	1023	1292	1294	1195
4	186	159	186	329	427	672	1262	1708	1413	1293
5	144	109	*298*	312	434	795	1510	**1618**	1693	1414
6	146	143	182	314	474	758	1297	2002	1891	1694
7	247	207	222	329	449	683	1066	1515	1463	1207
8	304	337	292	370	488	778	1026	1173	1168	905
9	252	330	333	408	512	683	844	966	942	841
10	225	254	357	397	533	598	680	833	813	735

Table B.4: The exact number of times each square was visited in 1000 runs of the danger region example by our system in runtime mode.

	A	B	C	D	E	F	G	H	I	J
1	485	459	412	431	582	661	687	858	861	771
2	370	380	418	453	565	690	779	929	971	934
3	343	378	363	423	565	746	978	1120	1122	1080
4	263	277	243	381	594	807	1184	1524	1410	1240
5	213	185	*262*	332	579	847	1405	**1496**	1681	1385
6	221	200	236	390	577	835	1212	1513	1379	1363
7	262	274	329	415	592	755	937	1045	1099	1161
8	299	354	390	457	573	652	812	922	966	997
9	332	346	371	436	528	618	702	803	842	868
10	491	427	411	447	524	650	688	774	780	829

Table B.5: The exact number of times each square was visited in 1000 runs of the danger region example by the comparison learner in runtime mode.

	A	B	C	D	E	F	G	H	I	J
1	-104	-146	-31	41	84	181	142	25	128	202
2	-22	-70	-13	50	87	71	109	140	380	262
3	-61	-104	-32	-23	-72	-104	45	172	172	115
4	-77	-118	-57	-52	-167	-135	78	184	3	53
5	-69	-76	*36*	-20	-145	-52	105	**122**	12	29
6	-75	-57	-54	-76	-103	-77	85	489	512	331
7	-15	-67	-107	-86	-143	-72	129	470	364	46
8	5	-17	-98	-87	-85	126	214	251	202	-92
9	-80	-16	-38	-28	-16	65	142	163	100	-27
10	-266	-173	-54	-50	9	-52	-8	59	33	-94

Table B.6: The difference of number of times each square was visited in 1000 runs of the danger region example by our system and the comparison learner in runtime mode.

B.3 Simple Safe Path Tables

	A	B	C	D	E	F	G	H	I	J
1	34	39	96	158	175	213	175	235	341	103
2	34	39	71	121	136	149	212	647	254	122
3	14	23	*179*	72	210	246	636	**5239**	637	255
4	37	75	253	166	271	323	1035	11311	1150	338
5	76	178	879	527	173	874	792	10158	654	229
6	38	72	1120	1410	1449	3608	3088	7228	962	172
7	34	58	165	339	290	960	344	1010	71	57
8	34	44	89	154	188	135	140	157	96	99
9	31	39	78	97	125	95	51	83	69	73
10	20	30	53	22	57	76	41	54	36	35

Table B.7: The exact number of times each square was visited in 1000 runs of the simple safe path example by our system in runtime mode.

	A	B	C	D	E	F	G	H	I	J
1	40	43	111	241	344	467	452	573	901	400
2	35	54	86	194	301	311	415	822	668	386
3	18	21	*39*	86	298	440	731	**1056**	844	619
4	24	19	38	119	323	490	643	1593	1010	1052
5	64	38	81	144	112	308	607	1400	949	1157
6	31	33	64	111	140	353	376	1025	779	946
7	23	43	44	49	167	354	493	724	311	322
8	25	35	32	80	170	141	298	388	299	391
9	26	31	65	114	188	172	157	310	328	292
10	21	59	68	52	164	237	155	288	194	200

Table B.8: The exact number of times each square was visited in 1000 runs of the simple safe path example by the comparison learner in runtime mode.

	A	B	C	D	E	F	G	H	I	J
1	-6	-4	-15	-83	-169	-254	-277	-338	-560	-297
2	-1	-15	-15	-73	-165	-162	-203	-175	-414	-264
3	-4	2	140	*-14*	-88	-194	-95	**4183**	-207	-364
4	13	56	215	47	-52	-167	392	9718	140	-714
5	12	140	798	383	61	566	185	8758	-295	-928
6	7	39	1056	1299	1309	3255	2712	6203	183	-774
7	11	15	121	290	123	606	-149	286	-240	-265
8	9	9	57	74	18	-6	-158	-231	-203	-292
9	5	8	13	-17	-63	-77	-106	-227	-259	-219
10	-1	-29	-15	-30	-107	-161	-114	-234	-158	-165

Table B.9: A comparison of number of times each square was visited in 1000 runs of the simple safe path example by our system and the comparison learner in runtime mode.

B.4 Random Safe Path Tables

	A	B	C	D	E	F	G	H	I	J
1	8	21	56	177	415	668	433	484	392	597
2	9	15	61	130	247	413	574	1866	958	860
3	7	7	*254*	117	139	384	1417	**2738**	1618	1097
4	13	26	168	95	189	346	788	1186	814	840
5	14	96	656	141	317	272	740	1053	406	536
6	9	80	719	259	522	325	550	881	285	177
7	9	28	112	37	222	159	265	345	299	133
8	35	43	34	55	142	162	175	176	200	90
9	38	50	29	59	70	121	99	121	135	121
10	17	37	56	87	86	151	69	55	105	93

Table B.10: The exact number of times each square was visited in 1000 runs of the safe path example by our system in runtime mode.

	A	B	C	D	E	F	G	H	I	J
1	36	37	83	111	243	427	397	384	368	821
2	35	73	118	158	204	313	588	1328	956	1579
3	21	27	*83*	131	173	396	1521	**1741**	1536	2015
4	26	35	87	125	232	356	861	791	1123	1997
5	44	95	144	190	330	294	782	703	598	1501
6	37	94	196	218	314	201	420	457	365	348
7	45	111	142	90	192	184	322	334	437	274
8	112	156	87	64	166	237	295	324	392	358
9	110	135	56	141	130	240	233	268	329	408
10	41	133	126	231	128	232	141	153	351	520

Table B.11: The exact number of times each square was visited in 1000 runs of the safe path example by the comparison learner in runtime mode.

	A	B	C	D	E	F	G	H	I	J
1	-28	-16	-27	66	172	241	36	100	24	-224
2	-26	-58	-57	-28	43	100	-14	538	2	-719
3	-14	-20	*171*	-14	-34	-12	-104	**997**	82	-918
-4	13	-9	81	-30	-43	-10	-73	395	-309	-1157
5	-30	1	512	-49	-13	-22	-42	350	-192	-965
6	-28	-14	523	41	208	124	130	424	-80	-171
7	-36	-83	-30	-53	30	-25	-57	11	-138	-141
8	-77	-113	-53	-9	-24	-75	-120	-148	-192	-268
9	-72	-85	-27	-82	-60	-119	-134	-147	-194	-287
10	-24	-96	-70	-144	-42	-81	-72	-98	-246	-427

Table B.12: The difference in the number of times each square was visited in 1000 runs of the safe path example by our system and the comparison learner in runtime mode.

B.5 Random Tables

	A	B	C	D	E	F	G	H	I	J
1	13	24	23	41	37	26	21	36	11	13
2	23	30	23	23	20	19	61	52	31	15
3	23	22	24	24	19	27	52	28	31	46
4	56	20	13	14	17	36	56	39	45	56
5	126	97	*521*	68	58	61	186	70	47	35
6	212	380	438	208	138	168	290	98	36	44
7	541	502	476	324	237	159	101	126	61	64
8	640	872	1310	706	432	246	145	115	70	61
9	1235	1892	2525	1165	384	232	301	145	95	82
10	766	**1522**	1825	2176	616	248	208	69	58	67

Table B.13: The exact number of times each square was visited in 1000 runs of the random world 1 by our system in runtime mode.

	A	B	C	D	E	F	G	H	I	J
1	309	229	71	27	51	55	55	34	14	19
2	279	214	139	102	72	63	77	46	20	13
3	167	113	84	76	44	58	62	48	36	23
4	469	233	76	49	40	36	42	33	35	24
5	687	374	*263*	134	113	75	102	59	40	18
6	835	818	581	366	184	162	123	54	40	24
7	1875	1135	689	436	262	188	75	79	53	35
8	2042	1565	1139	820	428	239	126	82	61	25
9	2869	2109	2041	1293	568	179	115	73	83	46
10	1972	**3282**	2664	1400	838	460	255	91	82	53

Table B.14: The exact number of times each square was visited in 1000 runs of the random world 1 by the comparison learner in runtime mode.

	A	B	C	D	E	F	G	H	I	J
1	-296	-205	-48	14	-14	-29	-34	2	-3	-6
2	-256	-184	-116	-79	-52	-44	-16	6	11	2
3	-144	-91	-60	-52	-25	-31	-10	-20	-5	23
4	-413	-213	-63	-35	-23	0	14	6	10	32
5	-561	-277	*258*	-66	-55	-14	84	11	7	17
6	-623	-438	-143	-158	-46	6	167	44	-4	20
7	-1334	-633	-213	-112	-25	-29	26	47	8	29
8	-1402	-693	171	-114	4	7	19	33	9	36
9	-1634	-217	484	-128	-184	53	186	72	12	36
10	-1206	**-1760**	-839	776	-222	-212	-47	-22	-24	14

Table B.15: The difference in the number of times each square was visited in 1000 runs of the random world 1 by our learner and the comparison learner in runtime mode.

	A	B	C	D	E	F	G	H	I	J
1	10	16	24	23	25	47	21	68	36	72
2	23	18	9	19	92	30	16	45	104	39
3	40	35	*110*	135	206	27	20	41	95	18
4	185	64	102	46	73	72	68	62	51	24
5	188	72	117	235	375	507	376	70	35	125
6	103	207	362	1909	3622	4419	2117	409	133	146
7	56	201	997	1021	**585**	1326	204	113	147	68
8	244	1991	6435	3520	1856	292	76	72	91	134
9	1059	2630	5367	1202	630	110	61	103	171	158
10	4543	5551	2838	504	186	73	213	164	557	118

Table B.16: The exact number of times each square was visited in 1000 runs of the random world 3 by our system in runtime mode.

	A	B	C	D	E	F	G	H	I	J
1	527	449	365	205	126	197	160	106	74	53
2	400	311	148	154	187	98	64	43	45	57
3	438	383	*261*	310	306	126	95	67	37	28
4	1055	846	368	375	270	277	166	50	32	37
5	1716	1285	639	515	**394**	431	319	91	71	84
6	1277	1196	799	862	1190	932	461	197	97	48
7	1080	960	832	957	710	907	376	203	59	53
8	724	1162	1073	1022	689	483	247	207	49	56
9	1351	1146	788	836	522	378	185	167	50	28
10	1411	645	575	580	466	266	159	161	44	17

Table B.17: The exact number of times each square was visited in 1000 runs of the random world 3 by the comparison learner in runtime mode.

	A	B	C	D	E	F	G	H	I	J
1	-517	-433	-341	-182	-101	-150	-139	-38	-38	19
2	-377	-293	-139	-135	-95	-68	-48	2	59	-18
3	-398	-348	*-151*	-175	-100	-99	-75	-26	58	-10
4	-870	-782	-266	-329	-197	-205	-98	12	19	-13
5	-1528	-1213	-522	-280	-19	76	57	-21	-36	41
6	-1174	-989	-437	1047	2432	3487	1656	212	36	98
7	-1024	-759	165	64	**-125**	419	-172	-90	88	15
8	-480	829	5362	2498	1167	-191	-171	-135	42	78
9	-292	1484	4579	366	108	-268	-124	-64	121	130
10	3132	4906	2263	-76	-280	-193	54	3	513	101

Table B.18: The difference of the number of times each square was visited in 1000 runs of the random world 3 by our system and the comparison learner in runtime mode.

	A	B	C	D	E	F	G	H	I	J
1	**1360**	1881	1942	1246	727	392	553	630	338	154
2	1182	1608	1845	1036	993	787	871	898	506	321
3	1046	1418	1383	1426	1234	632	336	210	133	350
4	1124	622	337	451	435	351	227	157	119	504
5	1452	524	349	323	221	198	295	222	49	106
6	1222	530	584	328	205	139	193	168	*136*	40
7	560	172	309	170	144	199	188	79	21	19
8	474	182	370	189	149	211	215	68	54	48
9	184	195	355	136	147	156	198	48	69	25
10	170	83	100	66	87	62	88	84	405	102

Table B.19: The exact number of times each square was visited in 1000 runs of the Random 4 World by our system in runtime mode.

	A	B	C	D	E	F	G	H	I	J
1	**2992**	4064	1856	1344	1827	986	785	830	621	205
2	3073	3111	2395	1147	928	866	438	465	423	245
3	2315	2188	1707	1093	894	593	178	169	153	151
4	1400	983	563	371	292	195	135	156	96	101
5	894	438	286	229	157	130	148	138	42	55
6	496	235	243	149	90	69	92	65	*16*	24
7	255	83	98	68	59	53	54	32	12	18
8	153	53	79	57	42	41	32	23	22	26
9	49	37	66	27	35	28	25	16	17	16
10	54	21	18	14	21	5	17	15	33	31

Table B.20: The exact number of times each square was visited in 1000 runs of the Random 4 World by the comparison learner in runtime mode.

	A	B	C	D	E	F	G	H	I	J
1	**-1632**	-2183	86	-98	-1100	-594	-232	-200	-283	-51
2	-1891	-1503	-550	-111	65	-79	433	433	83	76
3	-1269	-770	-324	333	340	39	158	41	-20	199
4	-276	-361	-226	80	143	156	92	1	23	403
5	558	86	63	94	64	68	147	84	7	51
6	726	295	341	179	115	70	101	103	*120*	16
7	305	89	211	102	85	146	134	47	9	1
8	321	129	291	132	107	170	183	45	32	22
9	135	158	289	109	112	128	173	32	52	9
10	116	62	82	52	66	57	71	69	372	71

Table B.21: The difference of number of times each square was visited in 1000 runs of the Random 4 World by our system the comparison learner in runtime mode.

Bibliography

Abbeel, P., M. Quigley, and A. Y. Ng (2006). Using inaccurate models in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pp. 1–8. ACM.

Argall, B. D., S. Chernova, M. Veloso, and B. Browning (2009). A survey of robot learning from demonstration. *Robotics and autonomous systems 57*(5), 469–483.

Atkeson, C. G. and S. Schaal (1997). Robot learning from demonstration. In *ICML*, Volume 97, pp. 12–20.

Bain, M. and C. Sammut (1996). A framework for behavioural cloning. In *Machine Intelligence 15, Intelligent Agents [St. Catherine's College, Oxford, July 1995]*, pp. 103–129. Oxford University.

Bellman, R. and R. Kalaba (1957). On the role of dynamic programming in statistical communication theory. *IRE Transactions on Information Theory 3*(3), 197–203.

Borenstein, J. and Y. Koren (1989). Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on systems, Man, and Cybernetics 19*(5), 1179–1187.

Borenstein, J. and Y. Koren (1990). Real-time obstacle avoidance for fast mobile robots in cluttered environments. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pp. 572–577. IEEE.

Brown, S. and C. Sammut (2013). *A relational approach to tool-use learning in robots*. Ph. D. thesis, University of New South Wales.

Brys, T., A. Harutyunyan, H. B. Suay, S. Chernova, M. E. Taylor, and A. Nowé (2015). Reinforcement learning from demonstration through shaping. In *IJCAI*, pp. 3352–3358.

Chiang, L. H., R. D. Braatz, and E. L. Russell (2001). *Fault detection and diagnosis in industrial systems*. Springer Science & Business Media.

Cichosz, P. and J. J. Mulawka (2016). Fast and efficient reinforcement learning with truncated temporal differences. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 99–107.

Dearden, R., N. Friedman, and S. Russell (1998). Bayesian q-learning. In *AAAI/IAAI*, pp. 761–768.

Dimitrakakis, C. and N. Tziortziotis (2013). Abc reinforcement learning. In *International Conference on Machine Learning*.

Geibel, P. (2001). Reinforcement learning with bounded risk. In *ICML*, pp. 162–169.

Geibel, P. and F. Wysotzki (2005). Risk-sensitive reinforcement learning applied to control under constraints. *J. Artif. Intell. Res.(JAIR) 24*, 81–108.

Greenwald, A., K. Hall, and R. Serrano (2003). Correlated q-learning. In *ICML*, Volume 3, pp. 242–249.

Heger, M. (1994). Consideration of risk in reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 105–111.

Huggins, J. and J. Tenenbaum (2015). Risk and regret of hierarchical bayesian learners. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1442–1451.

Jaksch, T., R. Ortner, and P. Auer (2010). Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research 11*.

Kadous, M. W., C. Sammut, and R. Sheh (2006). Autonomous traversal of rough terrain using behavioural cloning. In *The 3rd International Conference on Autonomous Robots and Agents*.

Koenig, S. and R. G. Simmons (1993). Complexity analysis of real-time reinforcement learning applied to nding shortest paths in deterministic domains. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 99–105. Citeseer.

Koenig, S. and R. G. Simmons (1994). Risk-sensitive planning with probabilistic decision graphs. In *Proceedings of the 4th international conference on principles of knowledge representation and reasoning*, pp. 363.

Kohri, T., K. Matsubayashi, and M. Tokoro (1997). An adaptive architecture for modular q-learning. In *International Joint Conference on Artificial Intelligence*, pp. 820–825.

Majumdar, A., S. Singh, A. Mandlekar, and M. Pavone (2017). Risk-sensitive inverse reinforcement learning via coherent risk models. In *Robotics: Science and Systems*.

Michie, D. (1998). Learning concepts from data. *Expert Systems with Applications 15*(3), 193–204.

Michie, D., M. Bain, and J. Hayes-Michie (1990). *Cognitive models from subcognitive skills*, pp. 71–99. Control, Robotics & Sensors. Institution of Engineering and Technology.

Mihatsch, O. and R. Neuneier (2002). Risk-sensitive reinforcement learning. *Machine learning 49*(2-3), 267–290.

Ng, A. Y., A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang (2006). Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*, pp. 363–372. Springer.

Sammut, C., S. Hurst, D. Kedzier, D. Michie, et al. (1992). Learning to fly. In *Proceedings of the ninth international workshop on Machine learning*, pp. 385–393.

Santara, A., A. Naik, B. Ravindran, D. Das, D. Mudigere, S. Avancha, and B. Kaul (2017). Rail: Risk-averse imitation learning. *arXiv preprint arXiv:1707.06658*.

Sheh, R. (2010). *Learning robot behaviours by observing and envisaging*. Ph. D. thesis, School of Computer Science and Engineering, The University of New South Wales.

Shen, Y., M. J. Tobia, T. Sommer, and K. Obermayer (2014). Risk-sensitive reinforcement learning. *Neural computation 26*(7), 1298–1328.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning 3*(1), 9–44.

Tsitsiklis, J. N. and B. Van Roy (1997). Analysis of temporal-diffference learning with function approximation. In *Advances in neural information processing systems*, pp. 1075–1081.

Van Moffaert, K., T. Brys, and A. Nowé (2015). Risk-sensitivity through multi-objective reinforcement learning. In *Evolutionary Computation (CEC), 2015 IEEE Congress on*, pp. 1746–1753. IEEE.

Watkins, C. J. and P. Dayan (1992). Q-learning. *Machine learning 8*(3-4), 279–292.